Mathematical Mappings for Identification Mechanisms

Von der Fakultät für Elektrotechnik, Informationstechnik, Physik
der Technischen Universität Carolo-Wilhelmina zu Braunschweig

zur Erlangung des Grades eines Doktors

der Ingenieurwissenschaften (Dr.-Ing.)

genehmigte Dissertation

von Saleh Mulhem
aus Damaskus.

eingereicht am: 02.12.2019
mündliche Prüfung am: 10.03.2020

1. Referent: Prof. Dr.-Ing. Wael Adi
2. Referent: Prof. Dr. Laurent Nana
3. Referentin: Prof. Dr. Anca Cristine Pascu

Druckjahr: 2020

For My family, Faika, and Azd

KURZFASSUNG

Sichere und physisch unklonbare Einheiten werden eine wichtige Rolle in künftigen Sicherheitssystemen einnehmen. Durch Einbettung solcher unklonbaren Objekte als integrale Systemkomponenten können zahlreiche Angriffsklassen auf moderne Netzwerke und Systeme verfolgt und verhindert werden. Es besteht daher ein dringender Bedarf an kostengünstigen und praktisch verwendbaren klonresistenten oder unklonbaren elektronischen und mechatronischen Einheiten.

Analoge physisch unklonbare Funktionen (PUFs) wurden vor zwei Jahrzehnten als DNA-ähnliche Materialeigenschaften entwickelt, die hauptsächlich in elektronischen Einheiten eingeführt wurden, um das Klonen dieser Einheiten zu erschweren. PUFs als analoge Einheiten leiden jedoch unter langzeitigen Inkonsistenzen aufgrund von Alterung, Betriebsbedingungen und anderen Faktoren. Dies führt zu einer Einschränkung der praktischen Verwendung von PUFs aufgrund ihrer hohen Komplexität und inhärenten Inkonsistenz.

Das Konzept der digitalen geheimen unbekannten Chiffre (SUC) wurde vor einem Jahrzehnt von einem Forschungsteam der Technischen Universität Braunschweig als kostengünstige und konsistente Alternative zur herkömmlichen analogen PUF eingeführt. SUCs wurden so konzipiert, dass sie sich während eines einzigartigen, unwiederholbaren Ereignisses innerhalb eines elektronischen Geräts eigenständig zu klonresistenten Modulen umwandeln. Die sich selbstgenerierenden Module stellen unbekannte operative Chiffren dar. Diese Chiffren sollen wie eine biologische DNA-Identität funktionieren, die schwer zu duplizieren und sicher identifizierbar sind. Das SUC-Konzept ist ein völlig neues technologiebasiertes Sicherheitsparadigma in der öffentlichen Literatur. Das neue Paradigma geht davon aus, dass die einzigen perfekten Geheimnisse diejenigen sind, die niemand kennt. Solche Geheimnisse können im praktischen Sinne nur durch physische invasive Angriffe oder durch eine umfassende Suche erreicht werden, die eine kryptografisch signifikante und unerreichbare Komplexität besitzen.

Um elektronische Einheiten klonresistent zu machen, sollten die erstellten SUCs nach dem einzigartigen Kreationsprozess dauerhaft und unveränderlich im Gerät bleiben. Deshalb müssen die SUCs in nichtflüchtigen, intelligenten und selbst rekonfigurierenden System-on-Chip-FPGA-Bausteinen generiert werden. Solche Geräte sind noch nicht verfügbar, aber es wird erwartet, dass sie in naher Zukunft auf den Markt kommen. SUCs haben als digitale Strukturen vernachlässigbare Alterungs- und Betriebseffekte während des gesamten Lebenszyklus einer Einheit.

Das Ziel dieser Dissertation ist die Entwicklung neuer spezieller Chiffrenklassen für SUCs mit Kardinalitäten von annähernd 2^{500} Chiffren. Die Verschlüsselungsstrukturen sind nicht konventionell, da nur neue ungewöhnliche FPGA-orientierte Funktionsmodule für die Abbildungsfunktionen verwendet werden. Insbesondere für die Verwendung der Mathblock-Core- und Zelleinheiten der Ziel-FPGAs sind die verwendeten Funktionen optimiert. Durch den Einsatz der speziellen FPGA-Ressourcen als Funktionskomponenten werden die SUCs mit geringer Komplexität und geringen Kosten für den praktischen Einsatz optimiert. In dieser Arbeit wurden viele neue nicht konventionelle Chiffre-Mapping-Strukturen gefunden und als SUC-Strukturen vorgestellt. Die Ergebnisse zeigen, dass in optimalen Fällen unbenutzte/tote Mathblocks „wiederbelebt“ werden können, um SUCs nahezu kostenlos zu erstellen. Es wird gezeigt, dass die erreichten Sicherheitsniveaus der SUCs skalierbar sind und den Post-Quanten-Kryptosystemen gewachsen sind. Solche SUCs sind in vielen zukünftigen Massenprodukten wie Fahrzeugeinheiten einsetzbar, um Klonangriffen entgegenzuwirken. Es werden mehrere SUC-Anwendungsprotokolle vorgestellt, die die höhere Effizienz dieser Technologie im Vergleich zur traditionellen PUF-Technologie belegen. Das Erstellen von SUCs ist eine sehr herausfordernde Aufgabe. Diese Arbeit ist ein Schritt in der fortlaufenden Forschung, die auf der Suche nach effizienteren intelligenten SUCs für zukünftige Anwendungen ist.

ABSTRACT

Secured physically unclonable entities are emerging to play major role in future security systems. Embedding such unclonable entities as integral system component allow to counteract and trace securely a large class of attacks on contemporary networks and systems. Therefore, there is a crying need for low-cost and practically usable and physically clone-resistant or unclonable electronic and/or mechatronic units.

Analog Physical Unclonable Functions (PUFs) were introduced two decades ago as DNA-like born units material properties mainly in electronic units to make them hard to clone. PUFs however, as analog entities, suffer from long-term inconsistency due to aging, operational conditions and other factors. This lead to limiting the practical use of PUFs due to their high-complexity and inherent inconsistency.

The digital Secret Unknown Cipher (SUC) concept was introduced a decade ago by a research team at the technical university of Braunschweig as a low-cost and consistent alternative to the conventional analog PUF. SUCs were concepted to be self-created in a single-event, non-repeatable process within electronic devices as clone-resistant modules. The modules represent self-created unknown operational ciphers which may serve as DNA-like biological identities which are hard to duplicate and are securely identifiable. The SUC-concept is an entirely new technology-based security paradigm in the public literature. The new paradigm assumes that *the only perfect secrets are those which nobody knows*. Such secrets in a practical sense, can only be reached by physical invasive attacks or by exhaustive search requiring infeasible complexity.

To make electronic devices physically clone resistant, the created SUCs should stays permanent and unchangeable after the single-event creation process within the device. Therefore, SUCs need to be created in non-volatile, smart and self-reconfiguring System-on-Chip FPGA devices. Such devices do not yet exist, however, are expected to emerge in the near future. As pure digital structures, SUCs are consistent in the whole device life-cycle.

The objective of this thesis is to develop new special ciphers-classes for SUCs with cardinalities approaching at least 2^{500} ciphers. The cipher structures are non-conventional as only new FPGA-oriented functional modules are to be involved in the mapping functions. In particular, the used functions are optimized to deploy Mathblock-cores and cell units of the target FPGAs. Deploying the special FPGA resources as mappings components, results with optimized low-complexity and low-cost SUCs for practical use. This work presents many new non-conventional cipher-mapping structures designed for that special FPGA environment. The results show that in optimal cases, unused/dead Mathblocks may be *"reanimated"* to create nearly zero-cost SUCs. The attained SUCs security levels are shown to be scalable to cope easily with the post-quantum security requirements. Such SUCs are usable in many future mass-products as vehicular units to counteract cloning attacks. Several SUC use-case protocols are presented showing the higher-efficiency of this technology compared with the traditional PUF technology.

Creating SUCs is a very challenging task. This work is a step in an ongoing research towards more efficient smart SUCs for future applications.

Contents

"*In the social jungle of human existence, there is no feeling of being alive without a sense of identity*."

Identity: youth and crisis.
Erik Erikson (1902-1994)

1. INTRODUCTION

Throughout the ages, the problem of identity has always been a cultural issue. Identity provides us with a fuzzy answer to the question "*who are we*?". The conceptual problem of this answer arises when the question becomes more precise: "How can we extract a unique identity from objects and organisms?". At this point, scientists and researchers have been able to contribute to the answer. The only measurable unique property, that is available in organisms, is deoxyribonucleic acid (DNA). DNA is the genetic code which generates itself autonomously with a high degree of biological uniqueness [1]. To answer the question *"who are we?"*, we have to look at DNA, as DNA is what identifies every living organism, including us, "*who we are*!".

In the last two centuries, several attempts to extract identities from objects, and organisms from their physical features have been carried out [2]. For instance, in 1888, Francis Galton took the first step in the field of "fingerprinting" and estimated the probability of two persons having the identical fingerprints [3]. Based on these results and research Sir William Herschel was the first to use fingerprints for identifications of people in 1916 [4]. This later on led to the field of "biometrics" which is still used till today to identify people.

But research did not stop making use of specific characteristics for the identification of humans. In the twentieth century, a random pattern in paper was developed for the identification of currency notes [2], and a random optical reflection pattern was proposed as

a reflective particle tag in [5]. But unlike living organisms that not only can be identified by their phenotype but also by their DNA, physical objects can only be described by physical characteristics that are not autonomously specific and unique. Furthermore, this also can easily lead to a replication of the given item without being able to distinguish between the original and the replicate. Due to this reason a transferal of the concept of DNA to electronic units/devices has recently been presented and several approaches have proposed in fabricating an identity for an electronic unit/device, that closely resembles the high confidence level of biological DNA.

In the last decades, several researches and studies have been published and presented the idea of identifying an electronic device by storing a secret key in an embedded non-volatile memory (NVM)[6] [7]. Unfortunately, such technologies don't provide sufficient level of security and fall short against physical attacks [8]. Nevertheless, the necessity to find an efficient solution for device authentication has motivated researchers to take different physical approaches, so that device identities can no longer be manipulated and cloned.

Physical Unclonable Functions (PUFs) were proposed to serve as unclonable identities for electronic devices [2]. Here, PUF offers a good alternative to a conventional non-volatile key storage [9]. PUF is defined as a DNA-like identity residing in each electronic device [1]. The core idea of PUFs is to create a physical function from unique properties or physical characteristics of an electronic device. The major challenges facing PUF-technology are due to the utilization of pure analog mappings, which are prone to noise, aging, metastability, sensitivity to temperature, supply voltage variations, and other factors [10]. As a result, most of the proposed PUFs still suffer from being too complex, costly to implement, and strongly affected by noise in emerging applications [11].

Achieving the dream of building an efficient physical identity for electronic devices using tamper-proof hardware would take a big leap into creating something that would closely resemble our DNA. But the problem in how is still open.

1.1. Physically Clone-Resistant Identity

As it is not simple to create a physical random/unclonable function from an electrical device the question is if that device itself can generate its own identity. In this case, the resulting identity is not only built based on physical characteristics but also based on a specifically designed algorithm. In [1], the device identity is created by triggering a single-event one-time process to generate/inject a permanent (unknown) function into a device with the help of a true random number generator (TRNG). The result of this mechanism is a device with a self-defined unknown identity, as a result of a so-called "electronic mutation" [12], that is verifiable and clone resistant. In other words, it is assumed that this "electronic mutation" emerges from a modern VLSI device which allows a self-creation of a permanent hardwired unknown secret function [1]. Thus, a secret unknown function is plausible and applicable with the help of a TRNG to be injected into a chip/device [13]. However, the realization of such mutations requires smart self-reconfiguration non-volatile Field Programmable Gate Arrays (FPGA) technology. The resulting secret unknown function

should be very hard to reverse, change or even remove and serves as a DNA-like physically clone-resistant identity for an electronic device.

Traditionally, there are two ways of cloning a device: physically cloning and constructing a predictive model. In the context of PUF, physically cloning means that the "unique" response of the PUF implementation can be successfully reproduced in another identical device [14]. On the other hand, building a predictive model (modeling attack) means that a constructed algorithm (especially, a Machine Learning algorithm) behaves indistinguishably from the PUF on all input/output pairs [15].

The physically clone-resistant identity is a newly proposed concept in the cryptographic engineering field. The physically clone-resistant concept indicates that if cloning a device requires seeking and recovering all secret random relevant device transactions (of order >> 2^{80}), the ability to clone will become close to impossible in most practical applications. In that case, the device would be physically clone resistant [16]. From a cryptographic point of view, generating a modeling-resistant function/identity requires that the created secret unknown function needs to fulfill the design requirements of a pseudorandom function. Otherwise, a predictive model of secret unknown functions can be built.

1.2. New Research Contributions

The main aim of this thesis is to devise new mathematical mappings as a first step towards creating a physically clone-resistant function/identity. Such a function can be appropriate for identification mechanisms. It has been proved that such a function needs to fulfil the necessary conditions of being a pseudorandom function, which results in the proposed function being resistant to modeling attacks or any type of distinguishing attacks. On the other hand, the key contribution of this thesis is to show how to use hard-core multipliers (Mathblocks) in modern FPGA devices to design a physically clone-resistant function/identity. Such multipliers are freely available for cryptographic use when not used for other device applications. The proposed functions classes are making use of such free multipliers to create good quality functions at very low cost. In particular, the proposed functions, so-called secret unknown functions/ciphers, are presented as practical alternatives to PUFs. A concept of secret unknown function/cipher is proposed as a pragmatic solution to counteract the PUF-weaknesses. The security level of such an unknown function/cipher is analyzed and evaluated based on state-of-the-art generic attacks such as distinguishing attacks and modeling attacks. Other types of attacks, such as physically cloning attacks and side-channel attack, are outside of the scope of this thesis and will not be considered as the real expected future hardware environment is not yet available.

This work started by replacing the core exclusive or operation (XOR) in a Feistel network with a new involution operation based on using multipliers. The resulting functions exhibit very reliable security properties and fulfill the design requirements of a pseudorandom function. The second major contribution was to develop new multiplier-based permutation polynomial functions to become involutions. The resulting permutation classes were deployed to serve as round functions in a key-alternating cipher design [17]. The exploitation

of the unused multipliers to design these permutations allows the achievement of practically very low complexity in the hardware.

In this thesis, the research contributions can be summarized as follows:

1. Presenting several state-of-the-art proposals of unknown functions based on PUFs and analyzing their drawbacks and vulnerabilities.
2. Presenting the concept of secret unknown functions/ciphers and building a simple mathematical model of such ciphers.
3. Designing new classes of Mathblocks-based ciphers for generating clone-resistant identities.
4. Designing generic protocols using clone-resistant identities to build a trust chain in large complex networks.

Based on these contributions, several generic protocols for creating a clone-resistant authenticated link between devices were designed and presented. The proposed protocols make use of the physically clone-resistant identity of each participating device. The main target of these protocols was to show the efficiency of trust-chaining in large networks when physically clone-resistant identities are deployed and involved.

1.3. Thesis Outlines

Figure 1-1 shows an overview of how the chapters relate to each other and how to read them. **Chapter 2** introduces the state-of-the-art basic concepts in the information theory and main definitions in cryptography required for this work. Mainly such as, fundamentals of block cipher, pseudorandom functions, etc. The introduced definitions and terminology will be used throughout the thesis.

The remainder of this thesis is composed of three main parts. The **first part** investigates the state-of-the-art foundations of unknown functions from different perspectives and discusses formal and conceptual issues in the proposed secret unknown functions/ciphers. The targeted implementation environment is presented, and the implementation strategy is carefully discussed.

In chapter 3 three proposals of the unknown functions are discussed as being closely related to the objectives of secret unknown functions/ciphers. Within the scope of this chapter, PUFs as usable unknown functions, PUF-based unknown key generation and block cipher deploying PUFs were reviewed and presented. The classification of security threats on these proposals were presented based on a systematic approach. This chapter has been published as parts of the introduction and state-of-the-art sections in the following papers:

- S. Mulhem *et al.*, “Low-Complexity Nonlinear Self-Inverse Permutation for Physically Clone-Resistant Identities”, Cryptography, 4(1), 6, 2020.
- S. Mulhem *et al.*, “New Mathblocks-Based Feistel-Like Ciphers for Creating Clone-Resistant FPGA Devices”, Cryptography 2019, Vol. 3, Page 28, vol. 3, no. 4, p. 28, Dec. 2019.

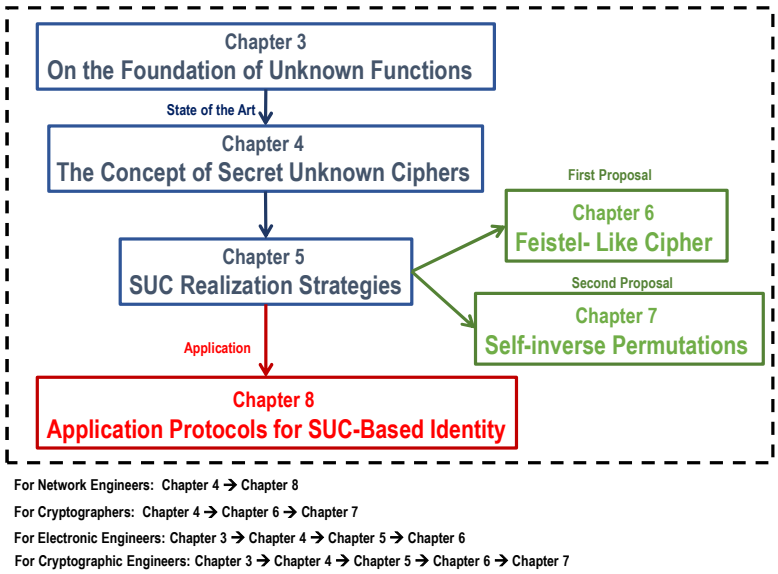


Figure 1-1. Argument Map for Thesis Outlines.

In chapter 4 secret unknown function/cipher was introduced as a new alternative to PUFs. This concept is considered as a Digital-PUF implemented in modern FPGA. Thus, the targeted implementation environment and its requirements were proposed and discussed. The conclusion of this chapter covered a simple mathematical model of secret unknown functions/ciphers and a new practical approach to evaluate the security of this model for post-quantum cryptography. This chapter has been published as parts of the following papers:

- S. Mulhem *et al.*, "New Mathblocks-Based Feistel-Like Ciphers for Creating Clone-Resistant FPGA Devices," Cryptography, no. Hardware Security, 2019.
- S. Mulhem *et al.*, "Security and Complexity Bounds of SUC-Based Physical Identity," in 2018 NASA/ESA Conference on Adaptive Hardware and Systems (AHS), 2018, pp. 317–322.

In chapter 5 a new technique for converting future smart programmable VLSI devices into physically hard-to-clone devices was devised. This technique allowed to generate new special huge classes of FPGA-optimized functions/ciphers. The targeted VLSI technologies are the smart self-reconfiguring and non-volatile FPGA devices for physically embedding the proposed structures. This cipher design strategy has been presented and published in the following papers:

- S. Mulhem *et al.*, "Low-Complexity Nonlinear Self-Inverse Permutation for Physically Clone-Resistant Identities", Cryptography, 4(1), 6, 2020.
- S. Mulhem *et al.,* "A New Low-Complexity Cipher Class for Clone-Resistant Identities," in 2019 42nd International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), 2019, pp. 971–976.

The **second part** of the thesis introduces two different new designs of secret unknown functions. In the first proposal, a Feistel-like cipher is proposed with two different proposals of implementation. The second proposal includes a new class of ciphers design based on the

so-called key-alternating cipher. Such ciphers use the proposed technique to convert future programmable VLSI devices into physically clone-resistant devices.

In chapter 6 the first proposal of a secret unknown function is introduced by replacing the exclusive or operation (XOR) in a Feistel network with a new involutive operation based on the multipliers. The most common threats against this proposal were discussed. Moreover, the inner function of the proposed Feistel-like cipher includes a new design structure based on only a few 4-bit mappings (golden S-Box). Such ciphers exhibit very low complexity in the hardware. This chapter has been published in the following papers:

- S. Mulhem *et al.*, "New Mathblocks-Based Feistel-Like Ciphers for Creating Clone-Resistant FPGA Devices", Cryptography 2019, Vol. 3, Page 28, vol. 3, no. 4, p. 28, Dec. 2019.
- S. Mulhem *et al.,* "A New Low-Complexity Cipher Class for Clone-Resistant Identities," in 2019 42nd International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), 2019, pp. 971–976.
- S. Mulhem *et al.*, "A Cipher Class Based on Golden S-Boxes for Creating Clone-Resistant Identities" in IOSes 2018 International workshop on Information & Operational Technology (IT & OT) security systems, Sep, 2018.

In chapter 7 the second proposal of a secret unknown function is presented as new large classes of permutations over a ring of integers based on T-Functions. The realization of the hardware/software complexities of the found classes were implemented in expected emerging non-volatile FPGA technologies. The hardware implementation results are provided and evaluated. This chapter has been published in the following paper:

- S. Mulhem *et al.*, "Low-Complexity Nonlinear Self-Inverse Permutation for Physically Clone-Resistant Identities", Cryptography, 4(1), 6, 2020.

The **third part** shows several primitive protocols for the generation of an authenticated network by deploying physically clone-resistant identities.

In chapter 8 several generic authentication protocols were discussed. The key result in this chapter is to show how to build trust relation between finite number of nodes/devices when one of them acts as a mediator node. Such technique did not have an influence on the proposed chain of nodes/devices. The main target of this chapter was to present the particular and efficient trust-chaining in large networks when the physically clone-resistant identity is involved. This chapter has been published in the following papers:

- S. Mulhem *et al.*, "Security and Complexity Bounds of SUC-Based Physical Identity," in 2018 NASA/ESA Conference on Adaptive Hardware and Systems (AHS), 2018, pp. 317–322.
- S. Mulhem *et al.,*"Chaining Trusted Links by Deploying Secured Physical Identities", Conference: Seventh IEEE International Conference on Emerging Security Technologies - EST September 2017, At Canterbury, UK.

In chapter 9 the main results and the important issues of the thesis are summarized, and some interesting future researches are suggested.

"*You should call it entropy, for two reasons. In the first place, your uncertainty function has been used in statistical mechanics under that name, so it already has a name. In the second place, and more important, no one really knows what entropy really is, so in a debate, you will always have the advantage.*"

"Scientific American", Vol. 225, (p. 180), 1971.

Claude Shannon (1916-2001)

2. BASIC CONCEPTS AND PRELIMINARIES

This chapter reviews the basic concepts in information theory and main definitions in cryptography such as block cipher, pseudorandom functions, etc.. These preliminaries are required for the later use throughout this thesis. Simultaneously, distinguishing security experiments are introduced as a main tool to evaluate the security of the designed cipher-classes in chapters 6 and 7.

2.1. Notations

The basic used notations throughout the thesis are presented as follows:

- F_{mn} denotes a set of all possible mappings from $\{0,1\}^m$ to $\{0,1\}^n$, where the cardinality of F_{mn} is $|F_{mn}| = 2^{n2^m}$.
- F_n denotes a set of all possible mappings from $\{0,1\}^n$ to $\{0,1\}^n$, where the cardinality of F_n is $|F_n| = 2^{n2^n}$.
- B_n denotes a set of all possible permutations (bijective mappings) from $\{0,1\}^n$ to $\{0,1\}^n$. Thus, $B_n \subset F_n$, where the cardinality of B_n is $|B_n| = 2^n!$.

- $f \xleftarrow{D} F_n$ denotes choosing the function f from F_n according to a probability distribution D over F_n.
- The uniform distribution is denoted by U.
- $f \xleftarrow{U} F_n$ or $f \leftarrow F_n$ denotes choosing the function f from F_n according to a uniform probability distribution U over F_n.
- $b \leftarrow \{0,1\}^n$ denotes randomly choosing the value b from $\{0,1\}^n$.
- $T(n)$ is a negligible function if it holds true that,

$$|T(n)| < \frac{1}{n^c} \quad , \text{ for every } \quad n>0 \tag{2-1}$$

Where, c is a positive real number. For instance, 2^{-n} , $2^{-\sqrt{n}}$, and $n^{-\log n}$ are considered good examples of negligible functions.

- $\mathbb{Z}_m$ denotes a ring of integers $\{0, 1, ..., m-1\}$ using addition and multiplication modulo m.
- $\oplus$ denotes a XOR operation over $\mathbb{Z}_2 = \{0,1\}$.
- log denotes logarithm base 2, where in this context the base of the log is assumed to be 2 resulting with the entropy being a number of bits.

2.2. Basic Concepts and Definitions

This section contains a survey of some basic mathematical concepts that will be employed throughout this thesis.

2.2.1. Block Cipher

There are, in general, two different categories of ciphers: Stream and Block ciphers. The main difference between them is that a stream cipher encrypts an arbitrary length of bit strings with memory, whilst a block cipher encrypts a limited length of bit strings without memory. In this thesis, the block cipher is considered as the backbone of this work and is defined as follows:

Definition.2.1: For fixed value (key) $K \in \{0,1\}^k$, a block cipher E is defined as a permutation over the input space $\{0,1\}^n$,

$$E : \{0,1\}^k \times \{0,1\}^n \rightarrow \{0,1\}^n \tag{2-2}$$

Where, $E(K,x)=y$ for every $x \in \{0,1\}^n$.

Note that each specific key selects one permutation from B_n. Therefore, a block cipher is basically a family of permutations with cardinality 2^k. Moreover, a permutation indexed by a secret key is used to process the enciphering of a plaintext in the encryption, while its inverse is used to process the deciphering of the ciphertext in the decryption. Both the encryption and decryption are efficiently computable given the key.

For instance, Feistel networks [18] [19] and Substitution-Permutation Networks (SPNs) are the most famous block cipher-structures, that were classified as two symmetric-key block ciphers using a secret key for both encryption and decryption processes. Several block ciphers have such structures. For instance, the most popular block ciphers are AES (Advanced Encryption Standard) with a different size of keys [20] and DES (the Data Encryption Standard) with a key size of 56 bits [21], where, AES design is based on SPN, while DES design is based on a Feistel network. The general design of a Feistel cipher and a SPN is illustrated in Figure 2-1 and Figure 2-2, respectively.

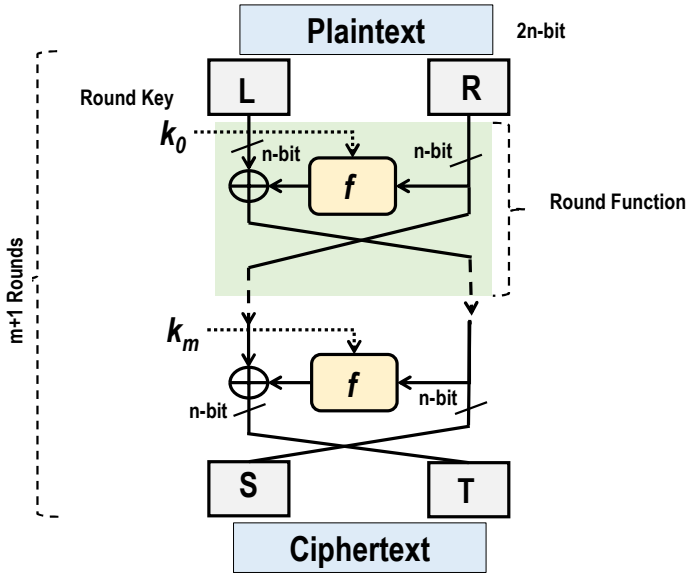


Figure 2-1. Sketch of Feistel Network.

In Figure 2-1, the construction details of a Feistel cipher is conducted where f is the (inner) round function, $k_0, k_1, \ldots, k_m$ are the keys of the rounds, and the basic operation is;

$$L_{i+1} = R_i \text{ and } \quad R_{i+1} = L_i \oplus f(R_i, K_i) \tag{2-3}$$

for each round $i=0,1,..,m$.

Furthermore, a block cipher using Feistel network as round function can be perceived as an involution. Therefore, Feistel network performs the encryption and decryption by using the same structure with reverse order of the keys for any (inner) round function f. To ensure high level of the security, the (inner) round function f should be bijection.

SPN produces the ciphertext after m iterated rounds. Each round is composed of Xor-operation, a substitution box (S-box) layer and a permutation layer as shown in Figure 2-2. The S-boxes should be bijection to ensure the invertibility. In particular, S-box layer and permutation layer should also fulfill the confusion and diffusion principles [22], where, the confusion refers to obscuring and hiding the relation between the plaintext, ciphertext and the secret key. While, the diffusion refers to spreading the influence of the plaintext bits and the secret key bits over the bulk of ciphertext. The advantage of Feistel ciphers comparing with SPNs is that the (inner) round function does not require being an invertible mapping.

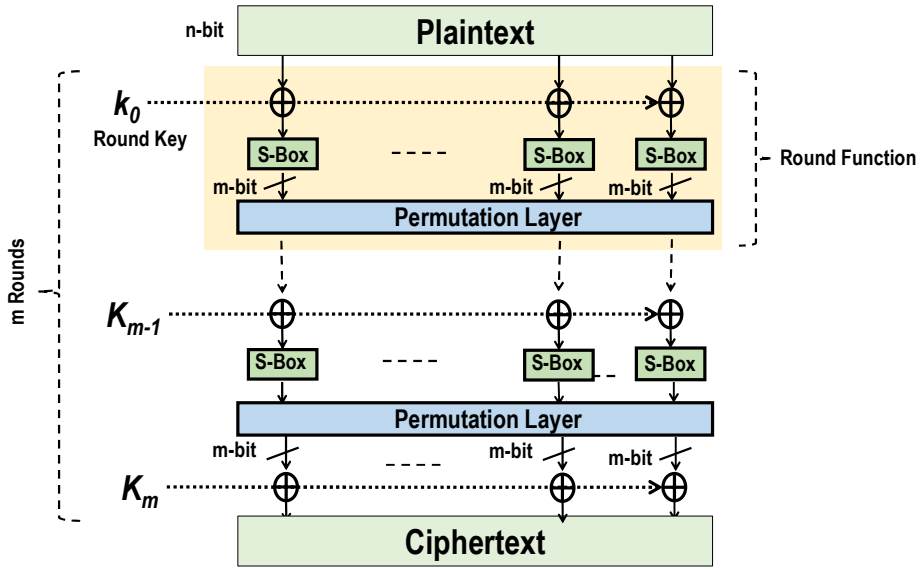


Figure 2-2. Sketch of Substitution-Permutation Network (SPN).

2.2.2. Pseudorandom Functions

In [23], Goldreich *et al.* presented the concept of a pseudorandom function (PRF). Several cryptographic applications of PRF were proposed in [24] such as dynamic hashing, constructing memoryless authentication scheme, etc.

Definition.2.2 [25]**:** A function f is a pseudo-random function (PRF) if it has two inputs such as $K \in \{0,1\}^k$ and an input data block $x \in \{0,1\}^m$, and one output data block $y = f(K, x)$, where $y \in \{0,1\}^n$. Here, for randomly chosen K, the function $f(K,\cdot)$ is a secure PRF if it "looks like" a random function from X to Y, i.e., it is impossible for an attacker (adversary) to link a given output to one of several known inputs.

Distinguishing Experiment-1:

Step 1: For a given PRF f, consider an adversary (distinguisher) Ψ that interacts with a challenger C who works as follows:

- C randomly chooses one bit $b \xleftarrow{U} \{0,1\}$.
- C returns $f \xleftarrow{U} F_n$ if b=1 to Ψ, otherwise returns $f \xleftarrow{D} f(K,\cdot)$, where $K \xleftarrow{U} \{0,1\}^k$.

Within time t.

Step 2: The adversary Ψ submits a polynomial number of queries (q) to challenger C.

Step 3: The adversary terminates the experiment by returning b'.

The advantage of Ψ to distinguish between $f \xleftarrow{U} F_n$ and $f \xleftarrow{D} f(K,\cdot)$, where $K \xleftarrow{U} \{0,1\}^k$, is defined as follows,

$$Adv_{PRF}^{f}(\Psi) = \left|\Pr[b = b'] - \tfrac{1}{2}\right| \tag{2-4}$$

And the maximum advantage over all Ψ is,

$$ADV_{PRF}^{f} = \max_{\Psi} \left| Adv_{PRF}^{f}(\Psi) \right| \tag{2-5}$$

Therefore, a PRF f is secure if ADV_{PRF}^{f} is negligible.

Theoretically, the quality of the PRF f affects a value of the advantage. For instance, if the advantage value is not negligible, then the adversary can gather more information about the function f. Therefore, a very weak PRF can be easily identified and their outputs could be predicted by a machine learning algorithm with little training. But when a PRF f is secure, the probability of the successful prediction of the function output becomes almost impossible.

To apply the distinguish experiment on a block cipher, some modifications are required. The challenger interacts with a set of all possible permutations B_n no more with F_n. However, the security level of such cipher could be analyzed after modifying the distinguish experiment as follows:

Distinguishing Experiment-2:

Step 1: For a given a block cipher E. Consider an adversary (distinguisher) Ψ that interacts with a challenger C who works as follows:

- C randomly chooses one bit $b \xleftarrow{U} \{0,1\}$.
- C returns $f \xleftarrow{U} B_n$ if b=1 to Ψ, otherwise returns $f \xleftarrow{D} E(K,\cdot)$, where $K \xleftarrow{U} \{0,1\}^k$.

Within time t.

Step 2: The adversary Ψ submits a polynomial number of queries (q) to challenger C.

Step 3: The adversary terminates the experiment by returning b'.

The advantage of Ψ to distinguish between a random permutation $f \xleftarrow{U} B_n$ and the block cipher $f \xleftarrow{D} E(K,\cdot)$, where $K \xleftarrow{U} \{0,1\}^k$, is defined as follows,

$$Adv_{PRP}^{E}(\Psi) = \left| \Pr[b = b'] - \tfrac{1}{2} \right| \tag{2-6}$$

As discussed in the distinguish experiment, a block cipher E is secure if any efficient adversary has a negligible value of the maximum advantage to distinguish E from a random permutation.

The following lemma provides a link between a secure block cipher and a secure PRF.

Lemma 2.1 [25]: **(PRF Switching Lemma)**

For a distinguishing experiment, let E be a block cipher defined over ($\{0,1\}^k$, $\{0,1\}^n$), Consider an adversary (distinguisher) Ψ that makes at most q queries to its challenger. Then,

$$\left| Adv_{PRP}^{E}(\Psi) - Adv_{PRF}^{E}(\Psi) \right| < q^2 / 2^{n+1} \tag{2-7}$$

Clearly, if any designed block cipher is a secure block cipher with very large input space 2^n, then this implies that this cipher is also a secure PRF.

2.2.3. Distinguishing vs Learning (Modeling)

The target in machine learning (ML) is to build a predictive model of an unknown function, algorithm, and/or concept by analyzing some training data [26]. For instance, a learner in the probably approximately correct (PAC) model receives samples in the form of $(x, f(x))$ as training data, and tries to make correct predictions of the output of the targeted function f [27]. Such a learning approach can be used as a ML algorithm for a cryptanalysis [28]. Therefore, if a learner L can predict the output of a PRF f based on past training data such as $(x_1, f(x_1)), \ldots, (x_q, f(x_q))$, then L can be utilized to distinguish the output of the targeted PRF f [26], and thus, the targeted function f is not a secure PRF.

Furthermore, a secure PRF concept postulates that the output of PRF f is statistically independent of $(x_1, f(x_1)), \ldots, (x_q, f(x_q))$ and uncorrelated to any learner [26]. Therefore, through this thesis, if a designed cipher is a secure PRF, then there is no ML algorithm that can build a predictive model for such a cipher.

2.2.4. Shannon Entropy

In information theory, the entropy indicates the measure of the amount of uncertainty involved in the outcome of a random process or variable. In other words, entropy quantifies the amount of the information, where, the rate of includable information in the outcome of a random process is directly proportional to the amount of uncertainty [22].

A system S abstractly has q distinguishable states, the amount of information $I(S)$ included in such system's state is determined as $\log(q)$. In particular, the occurring q distinguishable states are indicated by a given random variable X, and the self-information of measuring X is defined as,

$$I_X(x) = -\log[p_X(x)] \tag{2-8}$$

Where, $p_X(x)$ is a probability mass function of the random variable X with a value x (a specific value of the random variable X).

If $p_X(x)=1$, then $I_X(x)=0$. Furthermore, the expected value of $I_X(x)$ can define the Shannon entropy, where $\{x_1, \ldots, x_q\}$ are the possible values (states) of X, so that,

$$H(X) = E[I(X)] \tag{2-9}$$

And,

$$H(X) = -\sum_{i=1}^{q} p_X(x_i)\log_2[p_X(x_i)] \tag{2-10}$$

PART I

"*If you cannot solve the proposed problem, try to solve first some related problem. Could you imagine a more accessible related problem?*".

"How to Solve It". p114, Princeton University Press. 1957.

George Pólya (1887 - 1985)

3. ON THE FOUNDATION OF UNKNOWN FUNCTIONS

One of the essential security gaps in emerging electronic devices is the lack of a unique physically secure identity [13]. The need of such an identity is permanently growing to face the increasing threats to identification and authentication devices. Here physical unknown functions such as the Physical Unclonable Functions (PUFs) play an important role to fill up such gaps. The unknown functions are considered as a DNA-like identity for the electronic devices [16].

In this chapter, several state-of-the-art unknown function proposals are presented with some details as being closely related to the objectives of the proposed unknown crypto function. Parts of this chapter have been previously published in [29], [30], and [31].

The first basic approach towards creating a unique physically secure identity was presented in 2000 [32]. PUFs as a DNA-like identity have been proposed to provide the electronic devices with unique signatures/fingerprints through the born fabrication differences [1]. PUFs can be utilized as identification structures to prohibit cloning of electronic devices as in [2] and [33], or as a memoryless key storage [34], or for intellectual property (IP) protection [35].

One loosely related proposal to this work, was to deploy PUFs as components in a cryptographic scheme such as a block cipher [36], [37]. The resulting function was called a physical unclonable pseudo-random function (PUPRF) [38]. Within the scope of this chapter the PUPRF will also be looked at closely.

The uses of unknown functions in the published literature, which are related to this work, can be summarized to be presented in the subsequent sections as follows:

- Firstly, PUFs as unknown functions.
- Then, PUF-based unknown key generation to simulate an unknown function, especially for PRF.
- Finally, a block cipher deploying PUFs.

3.1.PUF as Usable Unknown Function

PUF was first proposed as a physical one-way function in an optical environment [32]. Such PUF uses the speckled patterns of an optical medium from laser light [39]. In [40], controlled PUF was defined as a controlled physical random function. Later, several designs of PUF were proposed such as arbiter PUFs [34], ring oscillator PUFs [41], TERO-PUF [42] etc. Furthermore, PUFs can be perceived as a physically unclonable source of randomness. The random physical factors are initiated during manufacturing. These random and unpredictable properties provide a PUF-structure with a high level of obscurity, unknowingness making the PUF substantially impossible to be cloned [43]. Thus, even the manufacturer cannot produce two identical devices with the same identities.

Theoretically, PUFs were gradually defined, firstly, as a physical one-way function [32], then as a physical random function [40], and lastly as a physical unknown/secret function [16]. A formalization of a traditional PUF concept was discussed and given in [44]. Following this work, the description of PUF technology is firstly introduced as follows:

Definition. 2.1: Let Γ be a set of PUFs, and $PUF \in \Gamma$. Then, PUF is defined theoretically as a mapping, which is hard to invert, unpredictable, and derived from the random behavior of a complex physical object/device,

$$PUF : \{0,1\}^* \rightarrow \{0,1\}^* \tag{3-1}$$

Where, * is a Kleene star and $\{0, 1\}^*$ denotes the set of all possible binary strings with finite lengths. The unclonability of PUF is inherited form being composed of many random components [43]. Therefore, PUF could also be defined as a source of randomness that is physically unclonable.

The inputs of a PUF are usually called challenges and the corresponding outputs are called responses. Moreover, Challenge-Response Pair (CPR) is a combination of a challenge and its response, and the set of all Challenge-Response Pairs of such PUF is called a CR space. PUFs are classified into two categories based on the cardinality of CRPs space; (1) a weak PUF which produces a few CRPs such as a ring oscillator PUF [41], and SRAM PUF [45],

(2) a strong PUF which offers a large number of CRPs such as arbiter PUF [34], and optical PUF [32].

The category of the strong PUFs seems to be more useful to entity authentication. The basic PUF-based authentication protocol is usually performed during two phases. Firstly, the enrollment phase: a server enrolls a challenge C of a PUF of device A and gets the corresponding response R, and after which the server stores the CRPs in the database (DB) which is very large as depicted in Figure 3-1. Secondly, in the authentication phase, the device A receives, for example, a challenge C_{Ai} and reproduces response R_{Ai}' of the PUF. In this phase, device A is authentic, if and only if, the distance between the stored R_{Ai} and reproduced R_{Ai}' is smaller than a threshold T. Such a threshold T is determined by the used error-correction code algorithm. Note that the enrollment phase should be performed in a trusted environment [46] and the authentication phase, however, should be performed in an insecure environment.

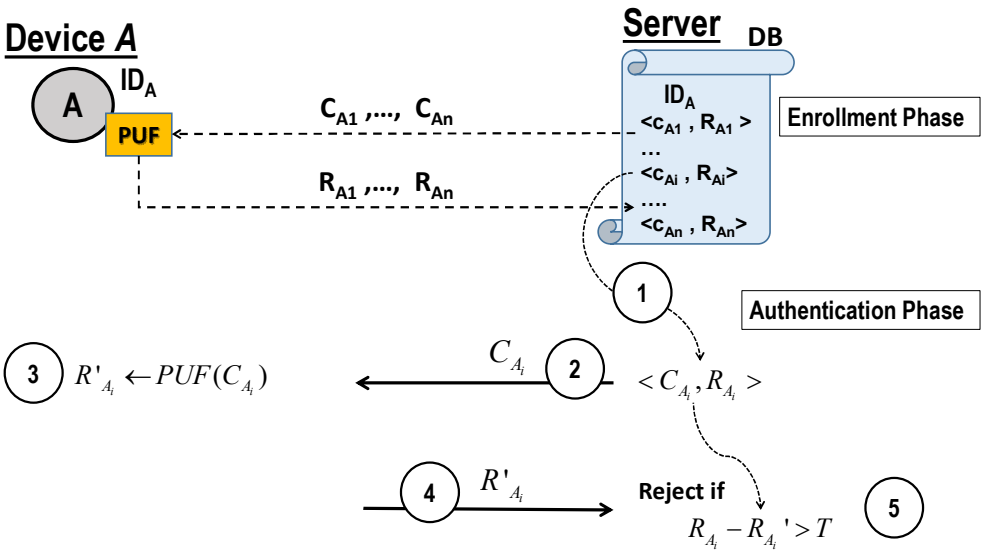


Figure 3-1. Basic PUF Authentication Protocol. Adapted from [10] and [47].

3.1.1. PUFs Security Drawbacks and Threats

In the following, PUF vulnerabilities and drawbacks are listed, and PUF threats and possible attacks are discussed. Two examples are presented to illustrate the possible countermeasure against such drawbacks and threats.

A. PUF Drawbacks

Due to various environmental perturbations such as operation temperature, supply voltage, aging, noise, and other effects, the majority of PUFs suffer from noise and therefore produce inconsistent responses. Therefore, PUF vulnerabilities and drawbacks can be presented as follows [10]:

- PUF response bits are not perfectly reproducible.
- PUF response bits are non-uniformly distributed.
- PUF CR bits exhibit strong correlations.
- PUFs produce a limited number of CRPs.

As countermeasures against such drawbacks, complex fuzzy extractors deploying error correction coding (ECC) [11] or a Helper Data Algorithm (HDA) were proposed to stabilize the PUFs response [48] [49]. Such solutions are able to extract a consistent response from the PUF response [50]. The implementation results show that the stabilization process of noisy responses requires a high cost of hardware resources for implementing an ECC algorithm. In other words, such solution mechanisms are highly area-complex and costly [11], [48]. For instance, the proposed fuzzy extractor of [51] requires 5.1 times of the PUF's resources to produce more consistent responses. Therefore, PUF technologies require a relatively high complexity in managing identification protocols which have made them unacceptable in many applications especially in automotive and IoT environments.

B. Threats on PUFs and Possible Attacks:

Due to the PUF vulnerabilities and drawbacks, the description and classification of the security threats facing PUF technology were investigated in [10], [45], and [52]. A systematic approach of such threats was presented in [10]. The threats and attacks can be summarized as follows:

- **Modeling Attack using ML Algorithms**: If there is a correlation between CRPs, then, a set of CRPs can be given as a training set to a ML algorithm [52], which constructs a predictive model of the PUF [15] [53] [54]. The predictive models of various proposals of delay PUF [55] are given based on different ML techniques with error ratios less than 1% for Arbiter PUFs, 1% for XOR Arbiter PUFs, 1% for Lightweight Secure PUFs, 4.5 % for Feed Forward Arbiter PUFs, and less than 1% for Ring Oscillator PUFs [15]. Therefore, the correlation between CRPs should be negligible. Otherwise an adversary can exploit this vulnerability to threaten PUFs.
- **Modeling Attack via PUF-Codebook:** Several PUFs produce a limited number of CRPs [10]. For instance, Crossbar PUFs [56] and Ring Oscillator PUFs [39] produce a quadratic number of CRPs. This implies that an adversary can generate a codebook of the PUF containing a look-up table of all CRPs to imitate the PUF [53].
- **Physical Cloning attack (CA)**: cloning an entity indicates the ability of reproducing the same entity. In the sense of cryptographic engineering, cloning attacks on a PUF means successfully reproducing the "unique" response of the PUF implementation in another identical device [14]. As a concrete example, by using a Focused Ion Beam (FIB) circuit edit (CE), an identical response to that of the SRAM PUF can be produced in a fully-functioning second instance of the device [14].
- **Side-Channel Attack (SCA)**: In contrast to invasive physical attacks, SCA does not damage the PUF or leads to the destruction of the secret. Therefore, SCA is considered as a real threat against PUF [57]. SCA target is to leak secret information and to reveal sensitive bits of information such as a PUF architecture and a fuzzy extractor design. Several SCAs on fuzzy extractors were applied [58]

to extract the information derived from a PUF. For instance, successful SCAs were conducted on Arbiter PUF [59] and Ring Oscillator PUF [58].

The combination between these attacks has led to a new attack that threatens PUFs [60]. However, the effective defense against such threats needs to address all possible security flaws in the PUF technology. Otherwise, an adversary can exploit these vulnerabilities to threaten PUFs.

The following two examples are presented to illustrate some possible countermeasures against such threats and attacks. The focus is put on expanding PUF CR space and improving PUF against Modeling attacks, which are closely related to the objectives of this work's proposal of unknown functions in chapter 4.

C. Attempts for Expanding CR-Space of Conventional PUFs

In the following, one mechanism for expanding PUF-CR space is presented. It is based on the fact that the uniqueness of the weak PUFs response is very good and almost ideal but the CR space of weak PUFs needs to be expanded due to weak PUFs produce a small number of CRPs [10]. Therefore, the realization of a strong PUF from a weak PUF was investigated and presented in several studies such as [61], [62], and [63].

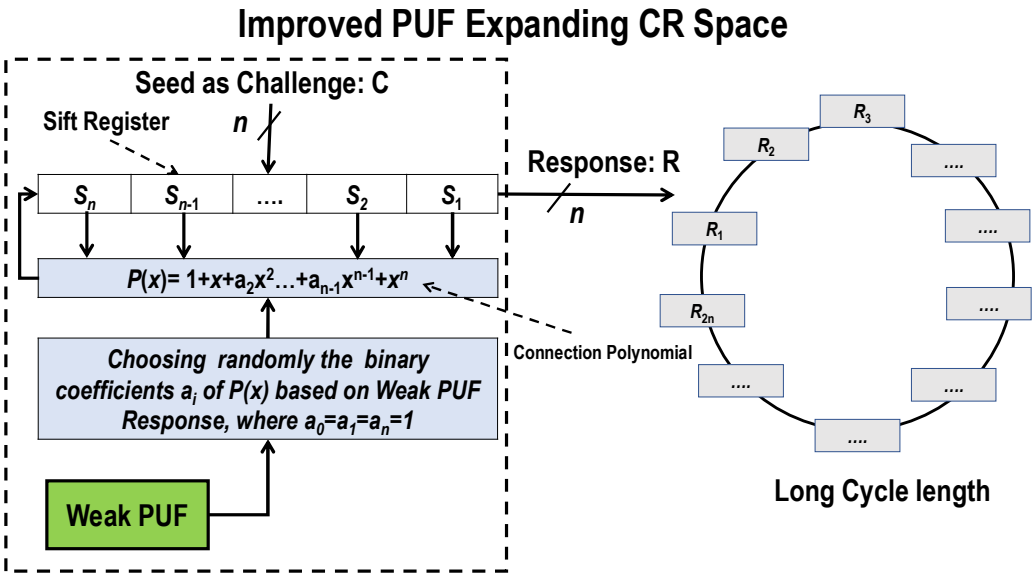


Figure 3-2. A LFSR-based Strong PUF Scheme through Deploying Weak PUF. Adapted from [63].

In [63], a reliable weak PUF as an entropy source and a linear-feedback shift register (LFSR) are deployed to construct a strong PUF. Figure 3-2 illustrates the structure of the LFSR-based strong PUF as an improvement of a weak PUF. The connection polynomial (feedback function) is randomly selected based on the weak PUF. Some rules for such a selection were proposed in [63] to make the output of the sift register have a maximum cycle length.

The proposed LFSR-based PUF produces an enormous number of responses. Therefore, it can be classified as a strong PUF. Unfortunately, the proposed LFSR-based strong PUF is not modeling attack resistance, as firstly, Berlekamp-Massey algorithm can be applied to

recover the selected connection polynomial and the register's seed [64]. Then, ML algorithm can build a predictive model for the weak PUF and later to the whole LFSR-based Strong PUF structure.

D. Reinforcing PUF Against Modeling Attacks

To make a PUF more robust, reliable, and having consistence responses, a controlled PUF was proposed in [40]. Figure 3-3 illustrates the concept of the controlled PUF as an improvement of a PUF. This is done by using control, where, a random hash function is utilized to prevent modeling attacks, and ECC is deployed to stabilize the noisy responses and return consistent responses. Note that the PUF is physically associated with a hash function. Therefore, any attempt to break or tamper with the link between the hash function and the PUF leads to the destruction of the controlled PUF circuit [2].

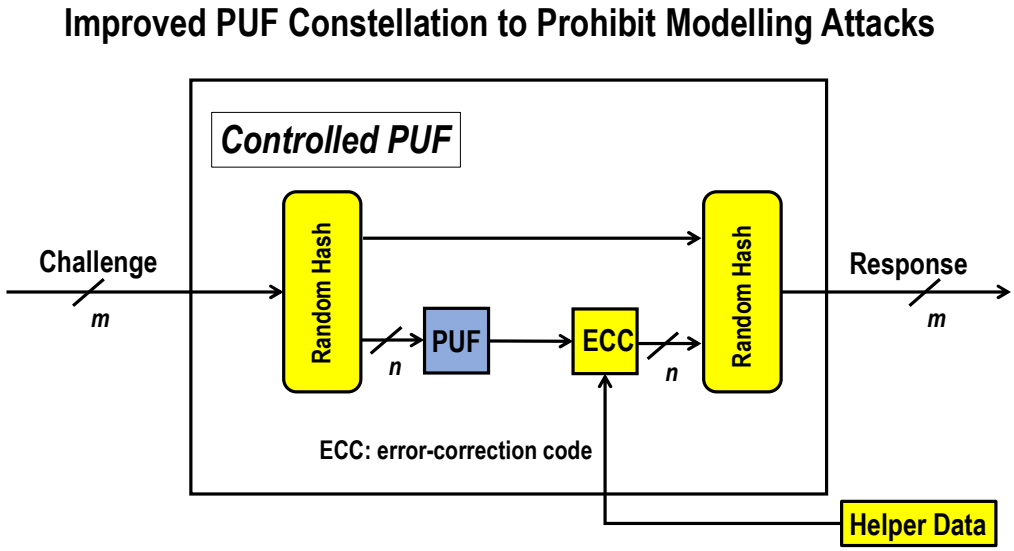


Figure 3-3. Controlled PUF Using Random Hash and ECC to Improve a PUF [40].

The controlled PUF structure deploys a PUF, especially a weak PUF, to randomize the output of the first hash function. This mechanism leads to increasing the number of all possible CRPs as well.

Furthermore, controlled PUF-based authentication protocol requires an additional step in the enrollment phase. This is done by the server storing a helper data p (HD) accompanied with each CRP. Figure 3-4 shows the authentication protocol using a controlled PUF.

(1) Server selects a triple $< c_{Ai}, r_{A_i}, p_{A_i} >$ from DB.

(2) Server sends (c_{Ai}, p_{A_i}) to the device A.

(3) Device A uses its controlled PUF to compute r'_{A_i}, or with more details, the controlled PUF uses a reproduction REP and a hash to return a consistent response r'_{A_i}.

(4) Device A sends back r'_{A_i} to Server.

(5) Server compares between the received response r'_{A_i} and the stored r_{A_i} and aborts if .

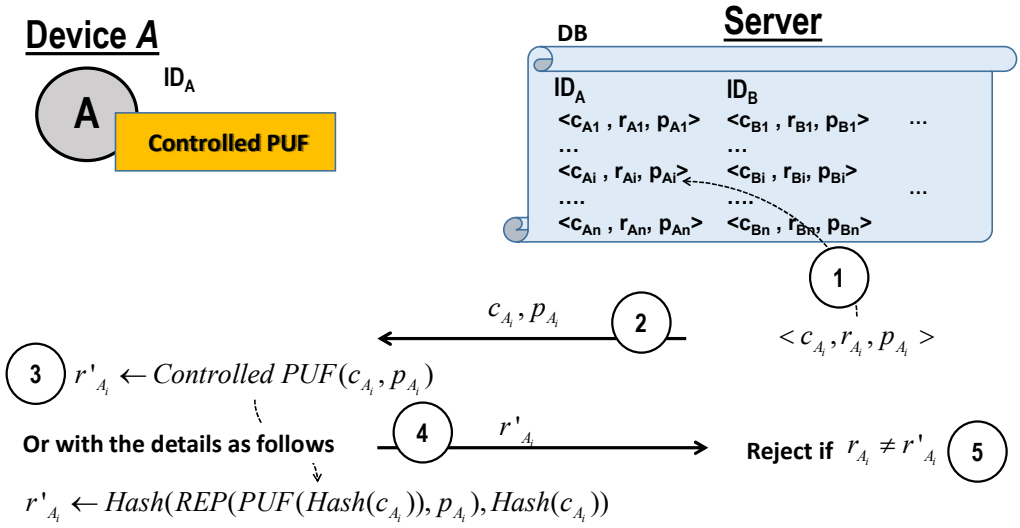


Figure 3-4. Device Authentication based on Controlled PUF. Adapted from [10].

After consuming all CRPs from device *A*'s record in DB, the PUF should return back to a secure environment for the update process of each device record. While the DB size linearly increases with the number of authentications [10], the number of CRPs of the controlled PUF, which depends on the original/silicon PUF, needs to expand its CR space.

However, the previous proposals require an expensive cryptographic hash function and/or a classic ECC algorithm for achieving an acceptable level of security, that put PUFs in conflict with the lightweight perspectives. Therefore, most of PUFs have the same difficulties in being complex and costly in utilizing the resources limited to IoT devices.

In conclusion, PUFs are recommended to be deployed for key generation [10], which seems more appropriate than other PUF-usages. On the other hand, some excellent surveys of various PUF designs and PUFs-based authentication protocols can be found in [2], [10] and [52]. However, still no breakthrough is expected towards attaining adequate traditional PUFs technologies for mass products.

3.1.2. PUFs Limitation, and Information Capacity Bounds

PUF was first defined as a physical one-way function (POWF) [32] and then as a random function [40]. At this point the question arises: does a PUF meet theoretically the criteria for being a one-way function and a random function?

The theoretical definition of the one-way function postulates that a function f from $\{0,1\}^*$ to $\{0,1\}^*$ is considered as one-way function, if it meets the following two conditions [65]:

- For every input x form $\{0,1\}^*$, $f(x)$ is easy to compute.
- $f^{-1}(x)$ is very hard to compute.

Where, $\{0, 1\}^*$ is the set of all possible binary strings with finite lengths. It is necessary to clarify that easy and hard should be seen from the perspective of computational complexity theory. In other words, f is one-way function if the probability of any

probabilistic polynomial-time algorithm to compute $f^{1}(x)$ is negligible, for all sufficiently large length of x. Therefore, the one-way function concept can only be taken into consideration when a function f with an infinite domain [44]. Therefore, it's no surprise then that PUFs *do not fulfil the requirements of one-way functions* as long as a PUF' domain is always finite i.e. it is it is always possible to invert a PUF by constructing an algorithm within a constant time bound as full look-up table for the PUF [44].

Here, PUF is defined as a mapping from a finite domain to a finite range as follows,

$$PUF : \{0,1\}^{m} \rightarrow \{0,1\}^{n} \tag{3-2}$$

And the existence of one-way function is still an open conjecture [65].

On the other hand, the theoretical definition of the physical random function postulates that a PUF from a set Γ is considered as a random function, if an adversary has a negligible advantage to predict a PUF- response to a random challenge [44]. Therefore, if the number of all possible PUFs $|\Gamma|$ is limited and the PUF-information capacity is very low, then the adversary has a good advantage to predict a PUF-response. This claim can be analyzed and discussed through the following approach:

Firstly, let F_{mn} be the set of all possible mappings from $\{0,1\}^{m}$ to $\{0,1\}^{n}$ having a cardinality,

$$|F_{mn}| = 2^{n \cdot 2^{m}} \tag{3-3}$$

And assume that a PUF is extracted from a physical object/system, the number of all possible produced PUFs is related to the entropy of the PUF-creation process, and the cardinality of a set of all possible produced PUFs is computed as 2^{H}, where, H is the PUF-creation entropy. In [44], PUF is considered as an isolated physical system S which fits into a sphere of radius R. The maximum entropy H_S (information content) of PUFs is upper bounded in its volume, as follows [44],

$$H_S \leq \alpha R^{2} \tag{3-4}$$

Where, α is a constant and defined as [66],

$$\alpha = \frac{\pi \cdot c^{3}}{\hbar \cdot G} \tag{3-5}$$

Where, c is the speed of light, G is Newton's constant, and $\hbar$ is the Planck constant. Eq(3-4) provides an upper bound for PUFs information content, and it indicates that the cardinality of the set of all possible produced PUFs is limited. Therefore, the set of all possible PUFs Γ is a subset of F_{mn}. Thus,

$$\Gamma \subset F_{mn} \Rightarrow |\Gamma| < |F_{mn}| \tag{3-6}$$

It is concluded that the number of all possible PUFs is limited and upper bounded.

Secondly, the following approach was proposed to determine the PUF-information capacity [38]. Here, PUF is defined as a silicon device (object) that implements a deterministic function [38], where, the silicon object consists of N silicon cells such as memory bits, flip-flops, latches, etc. Suppose that the delays values of these silicon cells are N continuous random variables having a normal distribution with a mean value T and variance σ_X^2. Because of noise found in silicon circuits, the capacity of information in one cell is represented as a continuous normal variable X with a variance σ_X^2. The capacity of information can then be determined as follows [67],

$$C = \frac{1}{2}\log_2\left(1+\frac{r^2\cdot\sigma_X^4}{\sigma_m^2(r\cdot\sigma_X^2+\sigma_m^2)}\right) \tag{3-7}$$

Where, r is the number of samples of X and m is a normal variable representing noise with a variance σ_m^2 [67]. In this case, the maximum information capacity of such PUF is given in [38] as,

$$I_{\max}(X_1,\cdots,X_N) \le N\cdot C \quad \text{bits.} \tag{3-8}$$

Therefore, the information capacity of PUFs is limited and small. Note that PUF-information capacity can be increased by increasing the number of silicon cells N. According (3-6) and (3-8), *a randomly chosen PUF from the subset Γ is not a random function*.

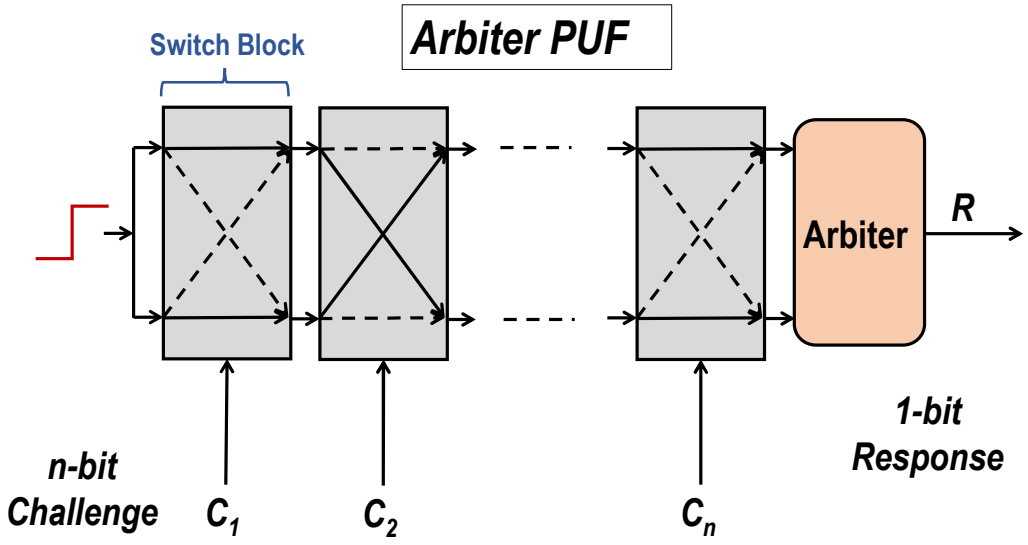


Figure 3-5. Hardware Sketch of an Arbiter PUF [2].

To make this point clearer, Figure 3-5 shows an example of a basic delay PUF circuit. Two paths cross a series of switch blocks (gates), the delay difference between the two paths determines a 1-bit response. In this case, a delay PUF is defined as a mapping of n-bit challenge and 1-bit response composed of n switch blocks. The number of all possible mappings from $\{0,1\}^n$ to $\{0,1\}$ is 2^{2^n}, but the question is still: *" are the n switch blocks able to produce 2^{2^n} delay PUFs ?"*. In [68], the results showed that the maximum number of all possible generated delay PUFs of size n is 2^{n^2} instead of 2^{2^n} different PUFs. Therefore, the set of all possible generated *delay PUFs* is a subset of the all possible mappings from $\{0,1\}^n$

to{0,1}. On the other hand, for n=64, the number of all possible generated *delay PUFs* is still very large number $2^{64^2} \approx 2^{4096}$. In this case, the information capacity of one *delay PUF* should be taken into account and determined. In [69], 64 switch blocks are deployed to construct an arbiter PUF. For 2^{64} different challenges, one arbiter PUF can respond by only one bit: either 1 or 0. Therefore, the information capacity of one arbiter PUF is very low, and brute force can be used to predict the arbiter PUF-response [53].

In conclusion, the number of all possible PUFs is limited and upper bounded. The information capacity of most PUF is low. Therefore, brute force attack and modeling attacks can be used to predict PUF-responses.

3.2. PUF-Based Unknown Key Generation for Pseudorandom Functions

The second approach of unknown functions can be perceived as a combination between PUFs and a cryptographic algorithm. This is the case when using an unknown key for a known cipher [70]. Where, a PUF-based key generation for a standard block cipher can be simply constructed by choosing a conventional cipher and taking the key source as the PUF unknown output/response K to that known cipher.

Figure 3-6 illustrates the concept of PUF-based unknown key generation. The resulting cipher with the unknown key K behaves precisely like a randomly selected invertible mapping from B_n. Therefore, the resulting cipher behavior fulfils the requirements to represent a secure PRF. As the cipher is public, the cardinality of all possible cipher structures using PUF-based unknown key generation is 2^m, where m is the size of the unknown key K.

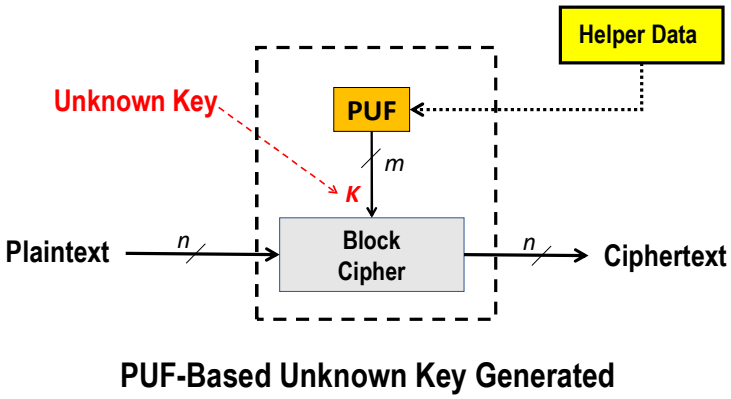


Figure 3-6. PUF-based Unknown Key Generation for a Block Cipher. Adapted from [70].

In [71], a generalized approach of PUF-based key generation for device authentication was investigated. Sadeghi *et al*. combined a PRF and strong PUF to achieve a high level of security in a protocol for RFID-system [71]. A strong PUF generates an unknown key to be

used for a PRF. The security evaluation was determined based on a PRF, but it was not related directly to a proposed strong PUF. However, this proposal achieves a high level of security, where, the resulting mapping response approaches a secure PRF behavior, and becomes hard to be impersonated or to be modeled. Unfortunately, the resulting mapping-structure still requires additional helper data to make the PUF respond with a consistent output (unknown key K).

3.2.1. Authentication Protocol via PUF-Based Unknown Key Generation

In [71], the proposed authentication protocol uses a strong PUF as a key generation of a PRF. The system exhibits a good level of security, where, the adversary has a negligible advantage to successfully attack the system.

Figure 3-7 illustrates the authentication protocol deploying PUF-based unknown key generation. The enrollment phase starts with initializing Z_0 as a random value inside a strong PUF to get the unknown K_A. The server then stores K_A of this strong PUF in DB. The authentication process is performed as follows:

(1) Server selects a random value a.
(2) Server sends a to device A.
(3) In the device A, NVM passes Z_0 to the strong PUF.
(4) The strong PUF computes K_A, where, K_A=$PUF(Z_0)$.
(5) Device A selects a random value b.
(6) Device A computes c by using E as $c = E_{K_A}(a,b)$.
(7) Device A sends $c||b$ to the server.
(8) Server selects the corresponding response K_A from DB.
(9) Finally, server rejects A if the received c is not equal to the computed $E_{K_A}(a,b)$ by the server.

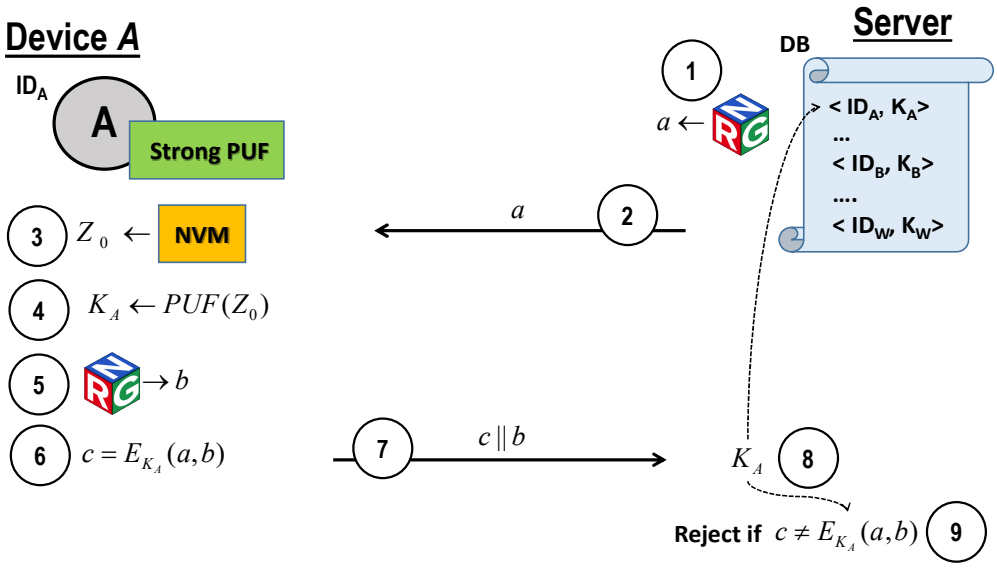


Figure 3-7. PUF-based Key Generation Usage for Device Authentication. Adapted from [71].

3.3. A Block Cipher Deploying PUFs as Unknown Function

The third possible approach can be described as an unknown cipher. The unknown cipher is an entirely new concept in the public literature. An unknown cipher can be created by using PUFs as a part of the cipher mappings. This approach can be constructed by combining a block cipher with a physical structure. The constructed function is called a physical unclonable pseudo-random function (PUPRF) [38], if it fulfills the requirements of PRF. Such a structure using diffusion and confusion principles is presented in [36] for specific applications such as software protection or device encryption.

3.3.1. A Block Cipher Deploying PUFs as Unknown Confusion Mappings

In [36], a block cipher deploying PUFs was proposed, where a cipher is constructed as a Feistel cipher rounds with PUFs as (inner) round functions. The resulting cipher fulfils the requirements of being a PRF, where, the resulting cipher is composed of three rounds of Feistel cipher deploying PUPRFs to generate a key.

Figure 3-8 illustrates the proposed block cipher deploying PUFs which is composed of a 3-ruond Feistel cipher utilizing 3 different PUFs as round functions, where, PUFs additionally need some HD for correct execution. To randomize the right part of the plaintext, ρ is selected from a random source. As parts of cipher-structure are unknown, the cardinality $|\Upsilon|$ of the all possible block ciphers deploying PUFs is computed as:

$$|\Upsilon| = \left(|\Gamma|\right)^3 << \left(2^{n\cdot 2^n}\right)^3 = 2^{3\cdot n\cdot 2^n} \tag{3-9}$$

Where, $|\Gamma|$ is the cardinality of all possible generated PUFs from $\{0,1\}^n$ to $\{0,1\}^n$. Although this resulting cipher exhibits a high level of security, the cipher-structure still requires a fuzzy extractor to ensure the consistency of the PUFs responses.

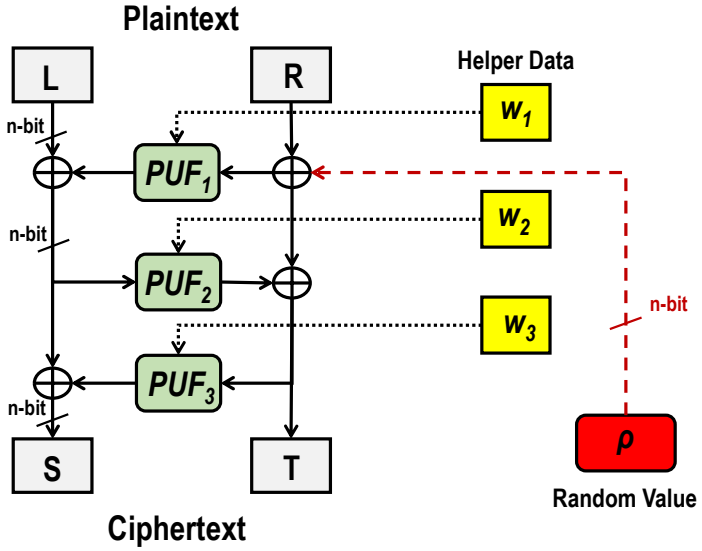


Figure 3-8. A Randomized 3-Round Feistel-Cipher Deploying PUFs. Adapted from [36].

3.3.2. Another Approach of a Block Cipher Deploying PUFs

A provable secure block cipher deploying PUFs was presented in [38]. This approach was introduced as 12 rounds of Feistel cipher with an SPN round function as depicted in Figure 3-9. The SPN round function consists of eight (4,4) PUFs instead of S-Boxes and a permutation layer. Note that a block cipher deploying PUFs is equivalent to a block cipher with secret components. Therefore, the security level of a block cipher deploying PUFs relates to the number of all possible generated PUFs.

Such proposal does not need to randomize any branch (L, R) of the cipher input. In the design of the inner function f, the permutation layer increases the diffusion property of the resulting cipher. Furthermore, a block cipher deploying PUFs may reduce the hardware cost and complexity by removing unnecessary cipher-key schedule/generator, but also requires a HD for each 4x4 PUFs to stabilize their responses in every round. The trade-off between removing key schedule and required HD needs more investigations.

In conclusion, although the block cipher deploying PUFs exhibits a high level of security, it still requires a fuzzy extractor to ensure the consistency of the PUFs responses. On the other hand, due to a PUF not being an invertible function, a block cipher deploying PUFs is restricted by the Feistel cipher structure. For instance, replacing the S-Boxes with (4,4) PURFs in SPN cipher does not makes it an invertible function. Therefore, a SPN cipher cannot be considered as a proposal of a block cipher deploying PUFs [38].

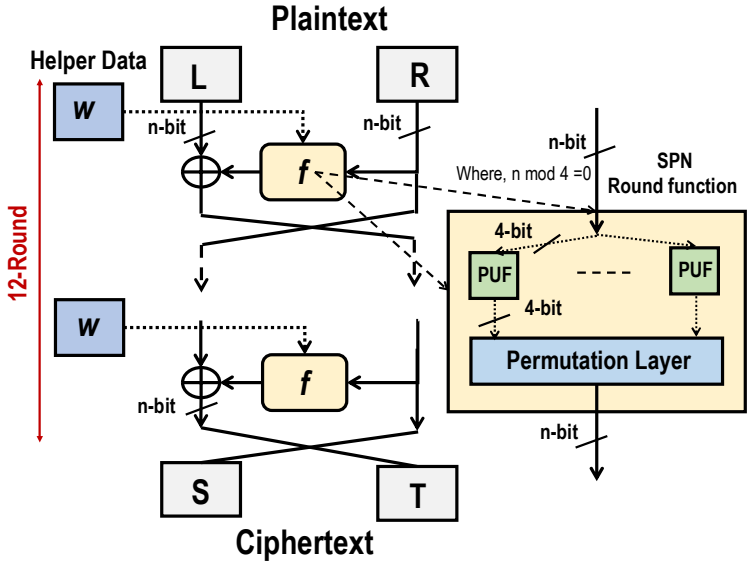


Figure 3-9. A Randomized 12-Round Feistel-Cipher Deploying PUFs. Adapted from [38].

3.4. Problems Motivating the Research Work

All previous proposals of unknown/random functions stay in conflict with the goal of attaining lightweight or low-cost usable cryptography. Unfortunately, no solutions in the published literature were able to counteract all the following PUF drawbacks so far:

1. **PUFs Inconsistency Problems:** Due to the noise, PUFs have inconsistent responses, which lead to the CRPs not being perfectly reproducible. To solve this

problem, complex fuzzy extractors using ECC or HD were traditionally proposed [11]. This solution requires extra cost in hardware and multiple communications with DB to run identification/authentication protocols. Other techniques were proposed as solutions to this structural problem [52]. For instance, adding additional primitive cryptographic tools such as a permutation, hash, etc., to the PUF circuit can stabilize the PUF responses and prevent modelling attacks [72], but these solutions are still costly in hardware [10].

2. **Expanding CR Space Problem:** Most PUFs cannot generate an enormous number of CRPs, therefore PUFs need to expand the CR space [73]. On the other hand, a block cipher based on PUFs solves this problem, as a block cipher with an input size n generates 2^n CRPs. Unfortunately, a block cipher based on PUFs still suffers from the same inconsistency problem as PUFs.
3. **CRP Management and Storage Problem:** The identification/authentication protocols based on PUFs require a huge DB to store all generated CRPs. In this case, the system requires extra loads to manage such DB. On the other hand, a mutual authentication protocol deploying a PUF was proposed in [74] as a solution to this problem. The suggested protocol stores only one pair in DB and updates it after every communication with a PUF device. Unfortunately, such a dynamic protocol requires a perfect PUF and it is not clear if this can prevent ML attacks [52].
4. **Limited PUFs Entropy for Identities:** As a PUF is considered to be as an isolated physical system or a silicon object this leads to the limitations in PUF-entropy.

In the following section, secret unknown functions/ciphers are presented as practical alternatives to PUFs. A concept of secret unknown functions/ciphers is proposed as a pragmatic solution to the above problems. The creating concept of a secret unknown cipher is presented. Simultaneously, a simplified model of such ciphers is introduced. The security of the resulting ciphers is analyzed based on generic attacks such as distinguishing attacks and modeling attacks. Other types of attacks such as CA and SCA are outside of the scope of this thesis and will not be considered as the future technology environment to be offered for realizing such ciphers is still not known.

"*Immediately the genie appeared and said to him "What wouldst thou have? I am ready to obey thee as thy slave; I and the other slaves of the lamp*."

The Arabian Night's Entertainment.
Anonymous (1881)

4. THE CONCEPT OF SECRET UNKNOWN CIPHERS

Secret Unknown Cipher (SUC) was first introduced a decade ago in [75] as a DNA-like identity for an electronic device. SUC is proposed as a self- created hardwired secret crypto function in a System on Chip (SoC) device by triggering a random single-event process [76]. The resulting SUC may serve as a digital PUF avoiding the drawbacks of conventional PUFs.

Furthermore, the unknown-cipher-concept is an entirely new security paradigm in the published literature. The unknown cipher here does not deal with protecting the communications or the links between at least two parties, a sender and a receiver, which requires the cipher to be common and known to both parties (Kerckhoffs's principle) [29]. Instead, SUC is fundamentally designed to serve as a clone-resistant identity for an electronic device [75]. Therefore, a cipher designed to be embedded (and used without the need to be known to anybody) as an unknown structure does not violate Kerckhoff's principle.

The aim of this chapter is to present the concept for creating SUCs as a new proposal of unknown crypto functions. This chapter encompasses several topics as follows:

- The SUC creation process.
- The investigation of a simple SUC model.
- The evaluation of a quantum brute-force attack on SUC.

The contents of this chapter have been published in [29], [30], and [31].

4.1. Creation Concept of Unknown Ciphers

Nowadays, smart devices, especially SoC devices, are offering new applications for emerging technologies. SoCs exhibit reconfiguring capabilities within the internal hardwired structures. It is expected that SoC devices will be capable of being a self-reconfiguring platform. Such a platform can dynamically reconfigure themselves. This feature is considered a revolutionary step in the SoC technology. Furthermore, some recent Field Programmable Gate Array (FPGA) devices offer self-partial reconfiguration together with a dynamic reconfiguration capability such as Xilinx [77] [78]. On the other hand, some FPGAs are capable of creating permanent internal structures, such as the non-volatile Microsemi FPGAs [79]. These technologies are expected to emerge and allow self-reconfiguring of permanent hardware structures in a non-reversible fashion in the near future.

In particular, a self-reconfigurable non-volatile SoC architecture can be deployed as a basic technology for creating (mutating) permanent hardwired secret crypto functions [16], that are considered as a DNA-like identity for such devices. These properties, features, and mechanisms together resemble biological mutations.

Therefore, self-reconfigurable SoC FPGAs provide a suitable environment to introduce a new crypto-design paradigm [75] which traditional cryptography does not have. The key idea behind secret crypto function creation is practically based on randomly self-creating unknown hardwired block ciphers SUCs. It is assumed that SUC would be realizable in emerging VLSI devices that allow self-creation of permanent unknown usable secret structures as “an electronic mutation” as indicated in [1].

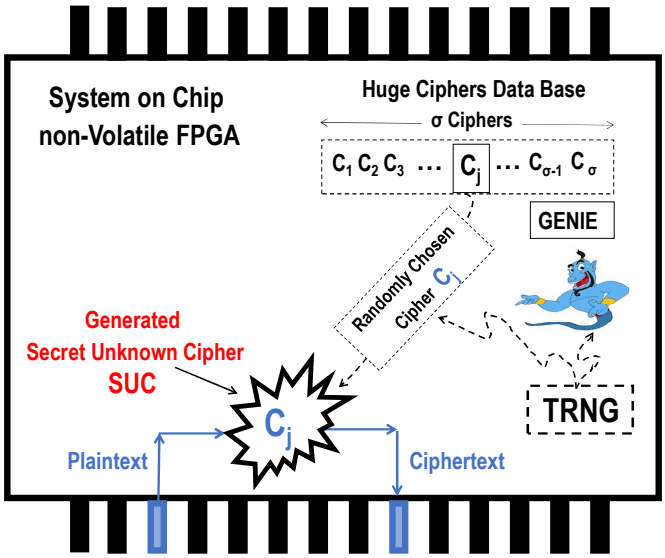


Figure 4-1. Key Idea for Generating a Secret Unknown Cipher [29].

Figure 4-1 illustrates SUC creation concept in a non-volatile FPGA device having internal self-reconfiguration capability and offering an internal true random number generator (TRNG), in particular:

1. Firstly, a huge class of distinct ciphers is generated such as $\{C_1, C_2 \ldots, C_\sigma\}$, where, $\sigma \to \infty$.
2. Following that, a one-time (single)-event process triggers TRNG leading to the random selection of one unknown cipher choice C_j from $\{C_1, C_2 \ldots, C_\sigma\}$.
3. Then, all dashed symbols and entities are fully and irreversibly deleted from the FPGA chip.

What remains inside the chip is only the non-repeatable unknown selected cipher C_j which is secret, and even unknown to the cipher designer himself, where, the designer has no influence on this process.

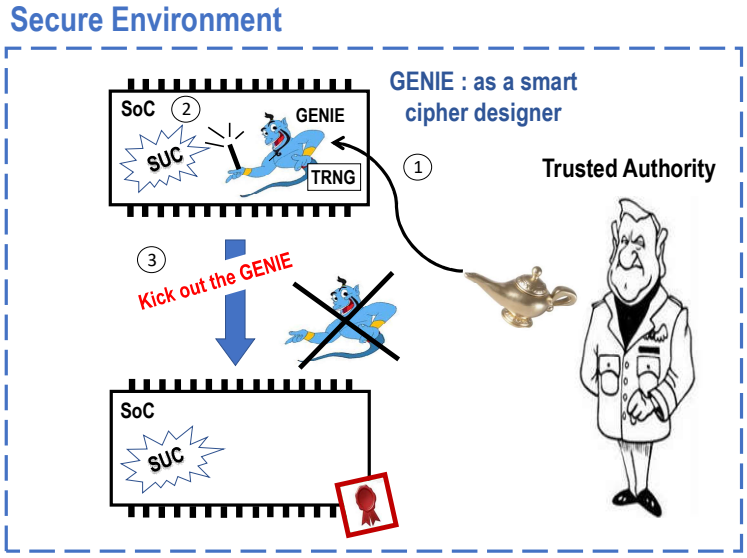


Figure 4-2. Mutating a Secret Unknown Cipher into SoC Device [29].

The proposed SUC is essentially launched based on the following fact: **"the only secret which can be kept unrevealed is the one which nobody knows**" [80]. Figure 4-2 illustrates the SUC creating process as follows:

1. Trusted Authority (TA) temporally injects in a single event a software package, a so-called "GENIE", into a SoC device. The GENIE is a deterministic algorithm that is defined as a smart cipher designer.
2. After being injected, the GENIE internally generates a permanent and unpredictable pseudorandom block cipher driven by the unknown and unpredictable TRNG bits.
3. After creating a SUC, the GENIE is completely and irreversibly deleted. What remains is an operational cipher (an SUC) which nobody knows.

It is worth mentioning that following other cryptographers who use the term Oracle (inspired by the gods) to describe a theoretical black box model, the term GENIE was inspired by the oriental folk tales of "One Thousand and One Nights". In the tales, the Genie is a supernatural creature which resides in a magic lamp and Aladdin, a poor youngster who finds the lamp, uses it to realize all his dreams. When Aladdin rubbed the magic lamp, the Genie immediately appeared and asked Aladdin about his wishes. Nobody knows how the

Genie grants wishes. In general, the holder of the lamp can wish almost anything, and the Genie will grant whatever it is.

This technique, using an internal TRNG, creates a single-non-repeatable and unpredictable SUC in the SoC device, which is described as:

$$SUC_t = GENIE(TRNG_t) \tag{4-1}$$

For every $t > 0$. This implies,

$$SUC_t : \{0,1\}^n \times \{0,1\}^{k_t} \rightarrow \{0,1\}^n \tag{4-2}$$

Where n is the bit size of the SUC input/output and k_t is the bit size of the cipher's secret key. In addition, SUC has the quality of generating a large distinct number of equally-secure Challenge/Response pairs as cleartext/ciphertext pairs, which is equal to 2^n. The reason is that CRPs are in that case cleartext/ciphertext pairs of a one-to-one mapping scanning the whole input and output space. *This overcomes the lack of challenge/response space in the case of PUFs* (see 3.4 Problem Motivating the research work- Expanding CRP space problem).

Furthermore, the created cipher SUC_t is operational and is unpredictable due to the $TRNG_t$ random sequence which is unknown to anybody. If $\{C_1, C_2 \ldots, C_\sigma\}$ is a huge class of distinct ciphers, then it is highly probable for any two-time points t_1 and t_2, that

$$TRNG_{t_1} \neq TRNG_{t_2} \Rightarrow GENIE(TRNG_{t_1}) \neq GENIE(TRNG_{t_2}) \Leftrightarrow SUC_{t_1} \neq SUC_{t_2} \tag{4-3}$$

In this case, each SoC device has its individual SUC with a probability approaching (1 - 1/σ) ≈ 100%, for sufficiently large σ > 0. Therefore, the GENIE can be perceived as a deterministic algorithm i.e. if TRNGs of two different SoC devices accidentally generates the same random value. Then, the GENIE will create identical SUCs in these devices. Such a possibility is equivalent to cloning an SUC which requires more investigation.

4.1.1. Targeted Technology and Platform Requirements

For an optimum security implementation of the self-created SUCs, the following technology requirements need to be met:

- The SoC device should be tamper-proof: Tamper-proof technology is used to store and process secret and private/sensitive information. Therefore, neither the users nor the adversaries with a physical access can tamper with it [81].
- SUC should be created in a single and unique non-repeatable event. Therefore, VLSI-technology, such as FPGAs with own hardware TRNGs, is required, which makes the resulting SUC unpredictable as a result of the random process [1]. This leads to the probability approaching zero that two equal SUCs can be generated in two different SoC devices.
- SUC should stay unreachable and permanently (non-removable) hardwired in a physical unit. Here, non-volatile VLSI-technology is required for long-term permanent storage and allows creating irreversible hardware locks when needed.

For instance, FPGA with a turned off bitstream or at least encrypted bitstream increases the difficulties of reaching the SUC-hardware design, and this makes the reverse engineering and Readback attack very arduous [82].

- Another requirement is to use the VLSI-technology in order to keep the SUC location random and unknown in the VLSI floorplan. The reason is that the adversary tries to extract information by probing some points inside the chip during physical attacks [82]. An increase in the number of possible probed points leads to an increase in the physical attack complexity. Therefore, the random location of each SUC makes successful physical attacks very challenging and almost infeasible.
- The cipher designer GENIE should be able to be temporally executable within the SoC device and easily irreversibly deleted.

Currently, off-the-shelf VLSI technologies do not completely fulfill all requirements as self-reconfiguring non-volatile FPGAs, although there is no technological reason for these not to be produced. The only technology which may fulfill the majority (but not all) of the previous requirements is Microsemi non-volatile SoC FPGA technology [79]. More details and information about Microsemi's non-Volatile SoC FPGA will be presented in chapter 5.

4.1.2. How to Enroll a SoC Device with Embedded SUC?

In the enrollment phase, a TA personalizes/enrolls the SoC_A in a secure environment as shown in Figure 4-3 The enrollment proceeds as follows:

1. TA randomly selects a set $\{x_1, \ldots\ x_T\}$ of cleartexts out of all 2^n possible pairs, where n is the bit size of the SUC input/output.
2. TA stimulates the SoC_A device to generate the corresponding ciphertexts by its SUC as a set: $\{y_1, \ldots y_T\}$.
3. The resulting T-(x_i, y_i) pairs are stored as secret pairs in the secured individuals device's records by TA for later use.

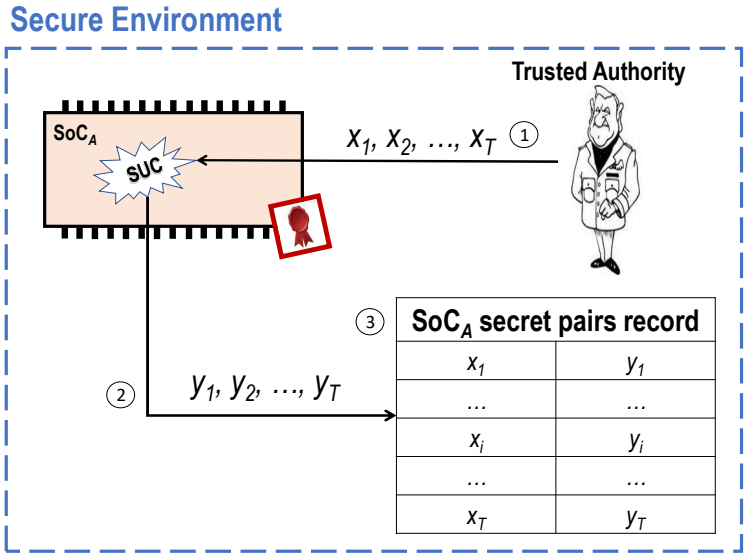


Figure 4-3. SUC Enrollment Phase in Secure Environment.

The randomly selected T-pairs represent a very small fraction out of the 2^n possible pairs. Note that the enrollment phase may be done independently by the same TA or many other TAs at a later time point depending on the use and application cases.

4.1.3. How to Use an SUC?

Figure 4- 4 shows a generic 2-way-identification protocol using SUC for authenticating a personalized SoC_A device which proceeds as follows:

1. A secret pair (x_i, y_i) is randomly chosen as a ticket from the TA's secret records of SoC_A. Then the TA challenges the SoC_A device by the cryptogram y_i over an insecure channel.
2. The SoC_A device responds by sending decrypted cleartext x_i'.
3. If $x_i' = x_i$, the SoC_A device is considered to be authentic. Then, being marked as used, the pair (x_i, y_i) will never be used again.

Another advantage of SUC technique is *that there is no need for multiple communications between a digital SUC and a server in order to attain a stable response comparing with the case of PUFs* (see 3.4 Problem Motivating the research work -Inconsistency PUF Problems).

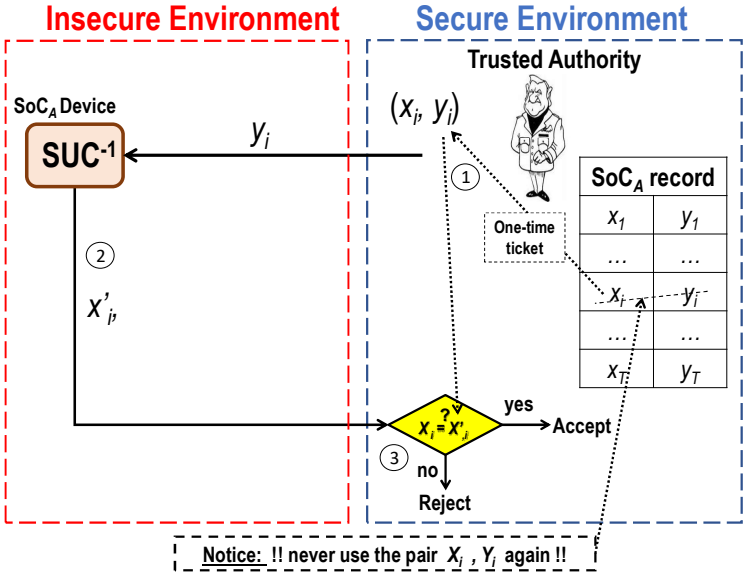


Figure 4-4. Two Way Identification Protocol over an Insecure Channel.

4.2. SUC- Mathematical Model

Mathematical models are generally methods of simulating the behavior of real devices and objects with mathematical equations/terms to forecast their future behavior. In the field of cryptography, the mathematical models cover the modelling of security properties of a designed system. For instance, the fundamental concepts in information security such as secrecy, authentication, and the notion of non-interference were discussed and modeled in [83] based on an algebraic approach.

Furthermore, the mathematical models do not only present a method to evaluate the security level of a system in the context of cryptographic engineering, but also allow to move

gradually from an abstraction level, towards implementation. In fact, this modelling approach gives rise to exposing the gaps in the system which might be noticeable by the adversaries. Thus, the SUC-Model can be classified under this context.

The proposed SUC-Model is composed of three levels: The first level presents the necessary definitions and notations. The second level presents the clone-resistant identity/device concept. In the third level, the combination between the previous levels allows to formalize SUC and determine the SUC-attack complexity. Figure 4-5 illustrates the SUC-Model that contains different levels, where the first and the second level are considered as foundation stones of the model.

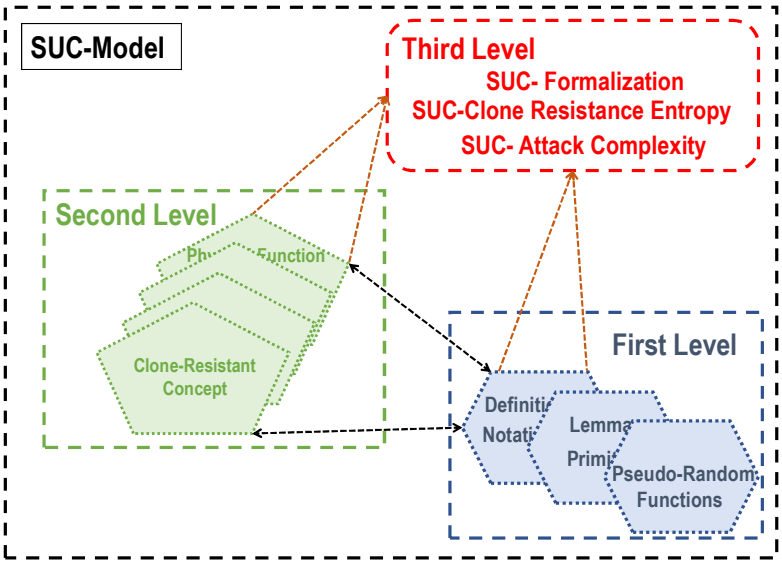


Figure 4-5. Blocks Hierarchy of SUC-Model.

4.2.1. Black Box Model as a Possible SUC-Attack Model

In cryptography, Kerckhoff's principle plays a fundamental role in dealing with ciphers that protect data transmissions between at least two parties, where, a sender and a receiver need the cipher to be common and known. Therefore, a cipher cannot be kept secret in the conventional cipher operation mode. Even though the cipher is kept secret in the SUC-Model, it does not violate Kerckhoff's principle, where, SUC is fundamentally designed to serve as a clone-resistant identity for a device [75]. On the other hand, a SUC should not be confused with "security by obscurity" [84], where, a cryptographer designs a cipher and then keeps the cipher secret and obscure. Therefore, the SUC-Model does not obey the security by obscurity as well as it does not contradict the Kerckhoff's principle in the classical cryptographic sense.

Thus, the SUC-Model can be perceived as a new paradigm in the cryptographic engineering field. However, the attacks on the SUC-Model is very similar to other PUF-proposals and obeys a black box model, since the adversary has only access to the input/output pairs. Therefore, the SUC attack model can be conducted considering the following points:

1. The adversary has no physical direct access to SUC internal design, but he can

only observe the input/output pairs behavior.

2. The SUC-attack module is defined based on a distinguishing attack in the sense of a black box model. Other scenarios such as differential cryptanalysis, linear cryptanalysis, and in a man-in-the-middle-attack (MIMA) require either exploiting the structure (internal design) of the cipher or have access to the cipher-round function.
3. From the hardware-attack perspective, a black box reverse engineering can be applied on SUC to reverse the chip internal design, the so-called Black Box attack. During the Black Box attack an adversary can generate some input combinations, while saving the corresponding outputs. The adversary then tries to extract the inner logic of the internal cipher design of the FPGA [85].

The flexibility of an adversary to gather and collect the SUC-input/output pairs determines the applicable attack scenarios. Thus, the SUC-attack model covers several attack scenarios such as [86]:

- Adaptive Chosen Plaintext Attack (CPA): The adversary can stimulate a SUC with any plaintext (query) and receive the ciphertext, after which he chooses the next plaintext depending on the previous input/output pair etc.
- Non-Adaptive Chosen Plaintext Attack (nCPA): The adversary chooses a set of SUC-plaintexts (queries) in advance.
- Adaptive Chosen Ciphertext Attack (CCA): The adversary can stimulate an SUC not only with any plaintext but also with any ciphertext as a query from both sides.
- Non-Adaptive Chosen Ciphertext Attack (nCCA): The adversary chooses a set of SUC-plaintexts or SUC-ciphertexts (queries) in advance.

Such scenarios classify the adversaries into two different categories: Adaptive adversaries and non-adaptive adversaries, where, CPA is considered the most general attack scenario for any adversary.

4.2.2. Basic Un-clonability and Modeling Attack Definitions

Cloning attack is one of the most important threats facing modern and emerging technology [14] [87]. In the case of PUFs, two types of cloning were defined as follows: Firstly, a physical clone, where a successful reproduction of the unique response of the PUF implementation into another identical device is applicable [14]. Secondly, modeling which is the construction of an algorithm (especially, ML algorithm) which behaves indistinguishably from the PUF on all CRPs [15]. Consequently, something can be physically cloned if its structure is known to somebody and something can be modeled if its input-output pairs behavior is predicted or distinguished by somebody. However, the unclonability of the SUC is deduced from the fact that nobody knows it and the SUC-Modeling resistance comes from the fact that a SUC is a PRF.

In the following, the SUC-cloning and -modeling resistances are discussed and presented.

A. SUC-Cloning Resistance:

Cloning only one SUC is generally equivalent to identifying and determining the targeted SUC out of all generated SUCs. In this case, the probability of a successful cloning of only one SUC is σ^{-1}, where, σ is the number of all possible generated SUCs. Therefore, the Cloning-Resistance Entropy (H_{CRE}) for only one SUC can be defined as:

$$H_{CRE} = \log_2(\sigma) \tag{4-4}$$

Note that the number of all possible SUCs σ grows exponentially with H_{CRE}. Therefore, if H_{CRE} is a very huge value, then cloning an SUC becomes almost infeasible.

Here, two different scenarios can be established:

The **first scenario**: If the GENIE is somehow kept secret, then the number of all generated SUCs σ attains the upper bound and H_{CRE} can be estimated as,

(4-5)

To estimate (4-5), Stirling formula of $m!$ is given as [88],

$$m! \sim \sqrt{2\pi m}\left(\frac{m}{e}\right)^m \tag{4-6}$$

And,

$$\ln(m!) = m\ln(m) - m + O(\ln(m)) \tag{4-7}$$

By choosing $m=2^n$, where $n>32$, and 2 as a base of the logarithm,

$$\log_2(2^n!) = n\cdot 2^n - 2^n\log_2(e) + O(n) \tag{4-8}$$

And,

$$2^n! = 2^{n\cdot 2^n}\cdot 2^{-2^n\log_2(e)+O(n)} = 2^{n\cdot 2^n - 2^n\log_2(e)+O(n)} \tag{4-9}$$

So that,

$$\sigma_{\max} = 2^{H_{CRE}} \approx 2^{n\cdot 2^n - 2.2^n} = 2^{2^n(n-2)} \tag{4-10}$$

In this scenario, the probability of successful cloning of an SUC is $\frac{1}{\sigma_{\max}} \approx \frac{1}{2^{2^n(n-2)}}$. Therefore, cloning an SUC is almost infeasible for $n>32$.

The **second scenario**: If the GENIE is published, then, the number of all generated SUCs is upper bounded by,

$$\sigma = 2^{H_{CRE}} < 2^n! \approx 2^{2^n(n\cdot -2)} \tag{4-11}$$

And H_{CRE} for only one SUC is upper bounded by,

$$H_{CRE} < 2^n(n-2) \tag{4-12}$$

In this scenario, the birthday attack is the best that any adversary can do to clone SUCs. According to the birthday attack, an adversary chooses v SUCs out of σ. Then, the adversary

randomly produces r SUCs and hopes that one of them matches one of v SUCs. Here, the expected number of successfully cloning of SUCs is given as follows [89],

$$\Omega = \frac{r \cdot v}{2^{H_{CRE}}} \tag{4-13}$$

The successful birthday attack on SUC is almost infeasible, if H_{CRE} satisfies,

$$H_{CRE} > \log_2\left(\frac{r \cdot v}{\Omega}\right) \tag{4-14}$$

If this condition is met, then the SUC is clone-resistant and the difficulty of physically cloning one SUC is close to impossible [16]. Furthermore, the result of the successful clone process should be a function that matches one of v SUCs. Therefore, increasing the cardinality σ for all generated SUCs is the most important condition to make the birthday attack on SUC very difficult.

B. SUC-Modeling Resistance:

When looking at modeling attacks on SUCs there are two possibilities: Firstly, the target of an adversary using ML is to create a predictive model of a SUC by analyzing some training data. Theoretically, if a SUC is a weak PRF, then certain patterns of plaintext/ciphertext pairs could be easily identified and detected by a ML algorithm with little training. But when a SUC is a secure PRF, the successful detection of patterns becomes impossible. Moreover, if a designed SUC is a secure PRF, then there is no ML algorithm that can build a predictive model for such a SUC, because the secure PRF concept postulates that the output of PRF is statistically independent of training data and uncorrelated with any learner [26].

The second possible modeling attack is to store all the possible plaintext/ciphertext pairs as the Cipher Codebook size CCBS=2^n. However, storing 2^n bits to build a model for a SUC is infeasible for ciphers with n>80.

In this thesis, the focus is put on the adversary who tries to use the collected SUC-input/output pairs in distinguishing attacks. Here, successful distinguishing attacks on SUCs indicate that the designed SUC structure is vulnerable. Therefore, sooner or later the adversary can build a predictive model for the designed SUC.

As a result, the self-generated SUC inside a chip can be modelled as a secure pseudorandom permutation (PRP) chosen randomly from $\{C_1, C_2 \ldots, C_\sigma\}$, where, $\sigma \leq 2^n!$ as

$$\begin{aligned} SUC &: \{0,1\}^n \times \{0,1\}^k \rightarrow \{0,1\}^n \\ (X, K) &\xrightarrow{PRP} Y \end{aligned} \tag{4-15}$$

Where, n and k are the input-output size and the key size, respectively. The inverse of SUC should be a secure PRP as well.

4.3. SUC vs Quantum Brute-Force Attack

In a traditional computer, a bit is considered a fundamental block that only has a state of 0 or 1. However, in a quantum computer the fundamental block is a so-called qubit [90] which can be in three states, 0, 1, and in both simultaneously, known as the super-position [91]. Note that any operation using a qubit in a super-position acts as 0 and 1 at the same time. Furthermore, two qubits can be entangled, therefore, if one of them changes a state, then the entangled qubit will change, where, the entangled qubit state is described as a single object with four different states [91]. Such properties lead to the exponential increase of the number of processed values in one operation [92].

Any classical search algorithm clearly requires, on average, $O(N)$ steps to specify an item in an unsorted database that contains N items. Surprisingly, the same problem can be solved in $\sqrt{N}$ steps by an algorithm using a quantum computer [93]. Consequently, any block cipher with a key size k can be cryptanalyzed/broken in time proportional to $\sqrt{2^k} = 2^{k/2}$ by using a quantum computer [25], where, the search space of such a key is 2^k.

In cryptography, Grover's algorithm is considered a special case of a more general search algorithm for quantum exhaustive search [25]. For instance, Grover's algorithm finds k_0 from the set K in $\sqrt{|K|}$ steps by querying a given function $\phi : K \to \{0,1\}$ defined as:

$$\phi(k) = \begin{cases} 1 & ; k = k_0 \\ 0 & ; k \neq k_0 \end{cases} \tag{4-16}$$

Where, $k_0 \in K$.

In the case of specifying one SUC from all possible SUCs, the search space size is given as $\sigma = 2^{H_{CRE}}$. Thus, Grover's algorithm requires to identify one SUC at least

$$2^{H_{CRE}/2} \text{ steps} \tag{4-17}$$

in the worst-case scenario (WCS).

4.4. Summary

In this chapter, the SUC-creation concept and process were presented and investigated as a new proposal for a usable unknown function. Here, SUC is proposed as a self-created hardwired secret function inside a chip. The results showed that selecting SUC from a class of ciphers with very high cardinality makes cloning attacks almost infeasible, and SUC-design should fulfil the requirements of PRFs to prohibit modeling attacks. Therefore, self-generated SUC inside a chip is modeled as a secure PRP randomly chosen from a huge class of ciphers.

"*A central lesson of science is that to understand complex issues (or even simple ones), we must try to free our minds of dogma and to guarantee the freedom to publish, to contradict, and to experiment. Arguments from authority are unacceptable*."

Billions and Billions: Thoughts on Life and Death at the Brink of the Millennium

Carl Sagan (1934-1996)

5. EFFICIENT SUC REALIZATION STRATEGIES

In this chapter, a new strategy of implementation is characterized for realizing the proposed SUC. This implementation strategy was introduced as one of the possible implementation scenarios. The contents of this chapter have been published in [29], [31], and [80].

The first step towards the digital system design started with Transistor-Transistor Logic (TTL) in 1970. The next step launched the gate arrays, which were established as a chip filled with NAND gates giving the designer the ability to interconnect these gates. The resulting logic design could serve as any logic function. From this seed, the programmability of different logic functions started. In 1980, AND-OR (gates) structures together with programmable connections were first introduced as Programmable Logic Arrays (PLAs). The most important improvement in the direction of programmability came with Programmable Logic Devices (PLDs), and later a collection of multiple PLDs, so-called Complex PLDs (CPLDs) [94]. FPGA is considered a very advanced step in this direction. It is composed of many programmable logic blocks connected through programmable interconnect network. This nature of FPGAs makes the programmable routing interconnect occupy 90% of the total area of the chip [95].

Generally, FPGAs, as depicted in Figure 5-1, consist of:

- Programmable/Configurable logic blocks (CLBs) that are utilized to implement the designed logic functions.
- Programmable routing which is used to connect the designed logic functions.
- Input/output blocks (I/O blocks).

FPGAs are classified and divided into three groups based on the storage technology. Firstly, SRAM-based FPGAs store the configuration data in volatile memory cells, such as Xilinx's 7-Series and Altera Stratix-5 [96]. Secondly, flash-memory-based devices offer a permanent data and structure storage. This essential property together with their programmability leads to nonvolatile Flash-based FPGAs, such as Microsemi SmartFusion®2 [79]. The third group of FPGAs contains anti-fuse FPGAs that are one-time programmable devices, as it is not able to return a burned anti-fuse into its initial state.

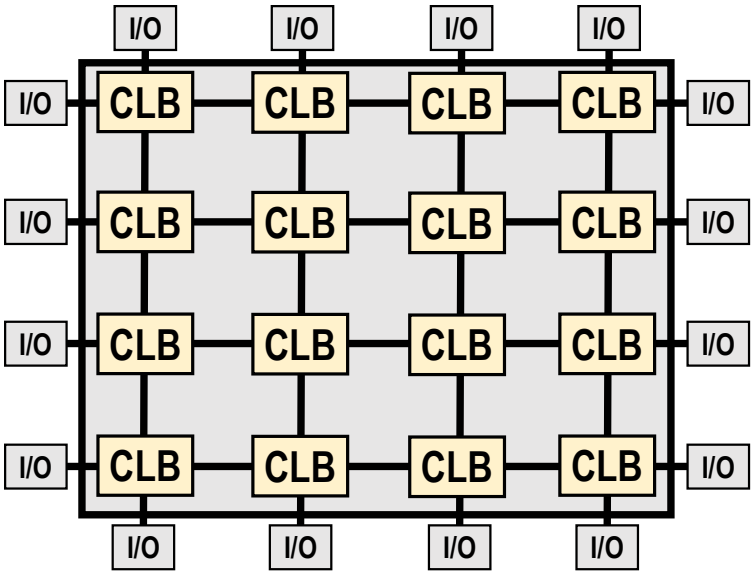


Figure 5-1. A Generalized Structure of an FPGA [95].

From a security point of view, each FPGAs family group has advantages and disadvantages. For instance, the volatility enables SRAM FPGAs to delete the data if it is tampered with [97]. Nonvolatility enables Flash-based FPGAs to permanently hold the necessary data, such as tamper logging and key revocation, through power-on/off cycles [96]. In addition, FPGAs allow the use of different cells and create different side channel behaviors [76]. Furthermore, the hierarchy of reconfigurable interconnects makes invasive attacks very difficult to penetrate into the internal hardware architectures. This increases the likelihood of the destruction of the secret itself during the attempt to reach or recover a secret embedded in the FPGA.

The non-volatile flash-based FPGA technology, such as SmartFusion®2 produced by Microsemi, is most suitable for the SUC-fabrication and embodiment. The FPGA flash fabric additionally incorporates an integrated ARM Cortex-M3 processor and contains integrated powerful arithmetic units, so-called Mathblocks, together with high-performance communication interfaces to all units of the chip. However, self-reconfiguration is still not possible in such devices. Therefore, the self-creation of permanent unknown "hard-wired"

structures as SUCs is still not possible in current products. Nevertheless, it is expected to be available for flash-based non-volatile technology in the near future. This will allow proposed mechanisms in this thesis to realize internal self-creating SUCs processes in such devices.

The greatest technology challenges in self-creating SUCs can be summarized in the following two categories:

1. Designing a GENIE program as a “smart VLSI-designer” which can incrementally extend an existing FPGA design without violating the technology design rules.
2. Designing a GENIE which can serve as an obedient “smart cipher creator” to fulfill all necessary security requirements.

There is no doubt that both challenges are highly complex. However, there are no technical reasons to believe that SUC creation will be an impossible mission. This work includes first new steps toward creating such SUCs.

5.1. SUC in Microsemi SmartFusion®2 SoC FPGA

SmartFusion®2 is one of the 4th Generation Flash FPGAs with a size of 65 nm non-volatile memory produced by Microsemi. SmartFusion®2 has the advantage of low power flash process, and many other features, such as a microcontroller (ARM Cortex-M3), 8 KB cache with very high-speed memory interfaces DDR2/3 controllers, and high-speed serial interface that can reach up to 5Gbps.

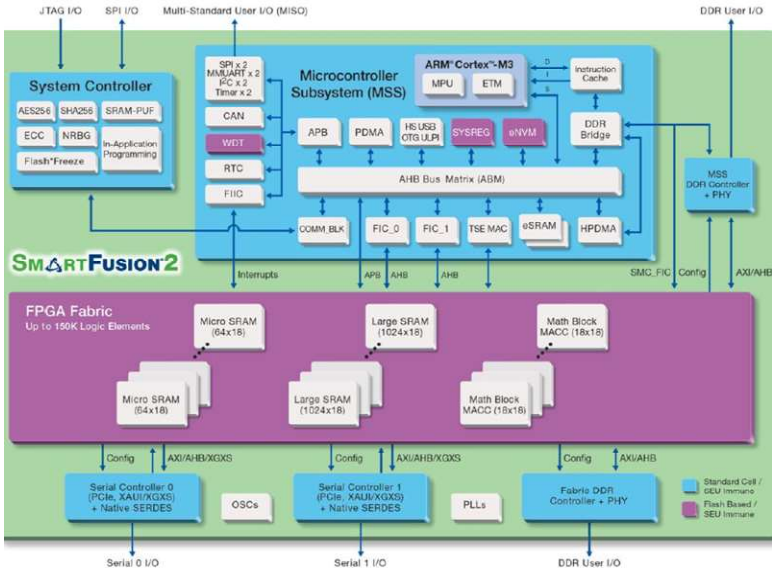


Figure 5-2. SmartFusion®2 SoC FPGA block diagram [79].

Figure 5-2 shows SmartFusion®2 SoC FPGA block diagram, which is composed of three main blocks [79]. The first block, in the upper-left-hand corner, is the System Controller, which contains some security cores, such as Flash Freeze, SHA-256, AES-256, RNG, and SRAM PUF. Therefore, SmartFusion®2 SoC FPGA is highly suitable for the targeted SUC security applications. The second block, in the upper-center, is the Microcontroller Subsystem (MSS) having a microcontroller ARM Cortex-M3. The third main block, near

the bottom, is the FPGA Fabric. This fabric contains an array of logic blocks, embedded hard blocks such as large static random-access memory (LSRAM), µSRAM, and Mathblocks. Here, the Mathblocks are distributed as rows inside the FPGA fabric.

Figure 5-3 illustrates a functional block diagram of every logic element, that is composed of a 4-input Look-Up-Table (LUT), a dedicated carry, and a separate D-flip-flop (D-FF).

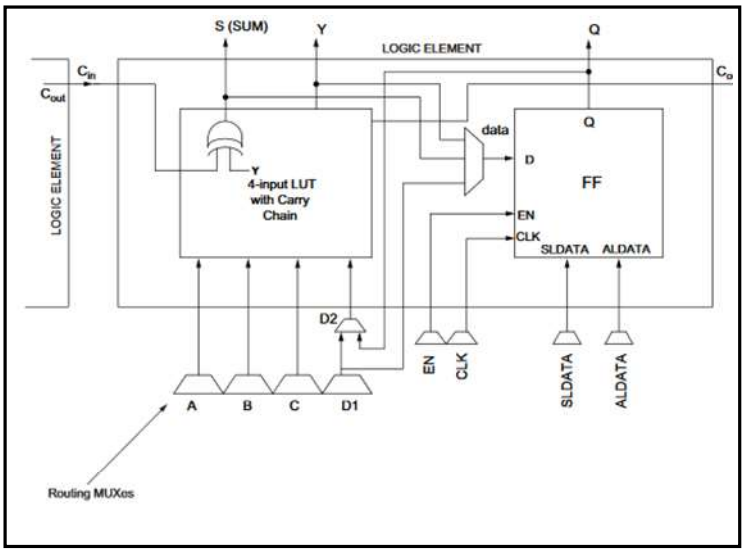


Figure 5-3. SmartFusion®2 FPGA Logic Element [79].

Note that *A*, *B*, *C*, and *D* are the LUT inputs, where, *Y* is the LUT output that can be XORed with carry input (C_{in}) to generate the S (sum) output. Therefore, the 4x1-LUT can be deployed to implement any 4x1 Boolean/arithmetic function. In addition, each set of 12 logic elements constitutes one physical layout cluster.

The family of SmartFusion®2 is large and contains different versions with several hardware capacities and features as shown in the Table I.

Table I. A Family of SmartFusion®2. Adapted from [79]

Feature	M2S005	M2S010	M2S025	M2S050	M2S060	M2S090	M2S150
# Maximum Logic Elements such as LUTS and DFFs	6060	12084	27696	56340	56520	86184	146124
# Mathblocks	11	22	34	72	72	84	240
Data Security	AES-256, SHA-256, and RNG	AES-256, SHA-256, and RNG	AES-256, SHA-256, and RNG	AES-256, SHA-256, and RNG	AES-256, SHA-256, ECC, PUF, and RNG	AES-256, SHA-256, ECC, PUF, and RNG	AES-256, SHA-256, ECC, PUF, and RNG

The data security feature is nothing else than a collection of cryptographic functions with RNG and SRAM PUF, that are used to protect the important information such as design IP.

5.1.1. SmartFusion®2 Mathblocks

SmartFusion®2 Mathblocks are embedded into the FPGA flash fabric. They are optimized for digital signal processing (DSP) applications such as Fast Fourier Transform. Furthermore, SmartFusion®2 Mathblocks have a built-in multiplier and adder, that can be used in cooperation with fabric logic to implement complex algorithms.

Each SmartFusion®2 Mathblock offers the following capabilities [79]:

- 18 x 18 bits signed multiplications.
- 17 x 17 bits unsigned multiplications.
- Dot product
- Built-in addition, subtraction, and accumulation units.
- Adder support: (A x B) + C or (A x B) + D or (A x B) + C + D.

SmartFusion®2 Mathblocks have two different operation modes of the multiplier, as shown in Figure 5-4, normal and DOT product modes. In the normal mode, a Mathblock performs the multiplication of two 18 bits inputs such as A[17:0] and B[17:0], to generate the output AxB of 36 bits. Moreover, Dot Product (DOTP) mode has two multipliers of 9-bit x 9-bit size with 36 bits adder, resulting with the following implemented equation:

$$DOT\ Output = \left(A[8:0] \times B[17:9] + A[17:9] \times B[8:0] \right) \times 2^{9} \qquad (5\text{-}1)$$

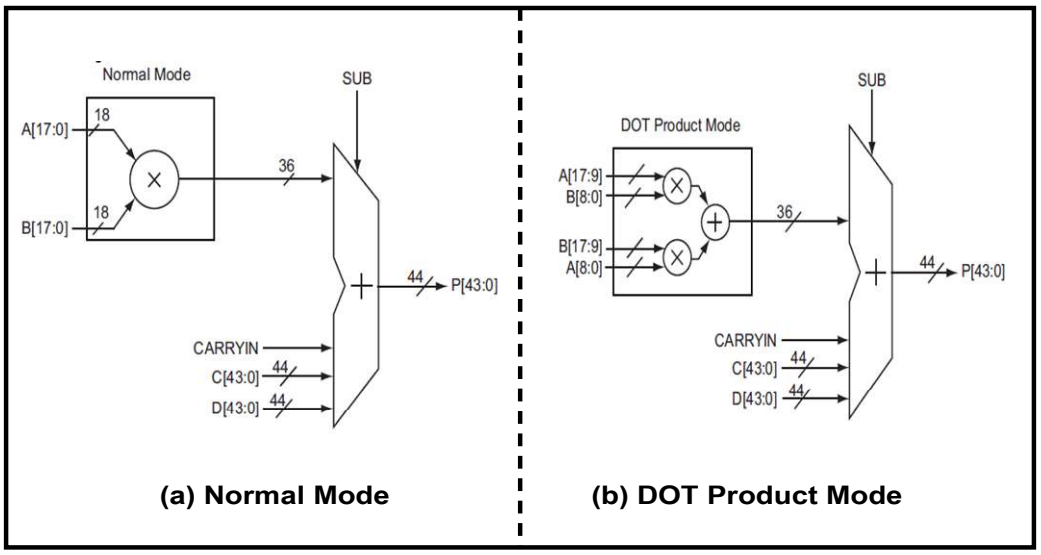


Figure 5-4. Mathblocks Diagram of (a) Normal Mode and (b) the DOTP Mode [79].

In both modes, the 36 bits output of the multiplier is added (subtracted) to C [43:0] input, CARRYING, and D [43:0] input, and the adder result is P [43:0]:

$$P[43:0] = \left(A[17:0] \times B[17:0] + C[43:0] + D[43:0] + CARRYING \right) \qquad (5\text{-}2)$$

Deploying Mathblocks as already available powerful multipliers in SUC design has the advantage of consuming much less amount of logic elements, which makes powerful SUC realization technically possible at low area cost on the FPGA.

5.2. SUC Design Strategy in SmartFusion®2

The biggest challenge of the SUC technique, is how to create a large class of good ciphers by a simple GENIE at acceptable cost in area and memory. Therefore, the main objective of this chapter is to approach strategies towards such targets.

Ordinarily, a good implementation strategy proposes the same ratio $R_{LUT/DFF}$ of LUTs and DFFs number, i.e., $R_{LUT/DFF}$ is close to 1 [98]. This results from the fact that most FPGAs architectures provide an easy to connect DFF with each LUT. Therefore, the desired SUC requires few LUTs and few DFFs in the targeted FPGA platform for lightweight implementation. The majority of standard cipher designs avoid deploying multipliers in the cipher mapping functions due to their high complexity. As such Mathblocks are often not completely used in many FPGA applications, it is wise to make use of such dead modules and reanimating them to implement usable security functions for free.

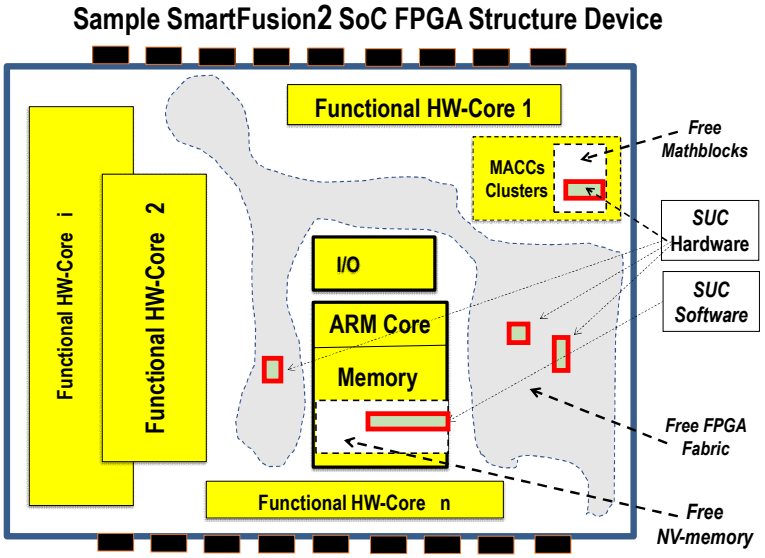


Figure 5-5. Sample Functional Layout After Creating a SUC in a FPGA Device [80].

The SUC realization strategies in summary are then:

- Using unconsumed FPGA resources such as the hardwired arithmetic addition and multiplication cores.
- Optimally utilizing the technology resources in hiding the SUC structure's keys and functions in permanent LUTs.

Figure 5-5 shows a possible scenario for an incremental embodiment "mutation" of a SUC in SmartFusion®2 SoC FPGA. In this scenario, the GENIE should use only resources outside the "functional HW-Cores", mainly consuming the non-used free Mathblocks resources, "free FPGA fabric", and "free NV memory". Ultimately, the SUC hardware and software components should also be distributed at individual locations in each mutated device.

The existing Mathblocks (MACCs), include multipliers which are optimized to be efficiently configured to perform 18x18 or double 9x9 multiplications. Here, the ring of integers $\mathbb{Z}_{2^n}$ can be adopted to avoid complex arithmetic. The realization of such multi-precision

arithmetic can be easily reached in hardware for a cipher block size n of 64, 72, 128, and 256 bits.

The existing FPGA resources should be taken into consideration, where many real applications do not consume all the available arithmetic cores. The SUC design strategy is therefore seeking ciphering functions for large cipher classes deploying mainly simple multiply, add and subtract arithmetic in $\mathbb{Z}_{2^n}$.

The most important requirements on the SUC cipher design is therefore to design huge classes of ciphers using multiplication as a major ciphering function.

PART II

"*Computer systems in general and personal "data banks" in particular need protection.*".

Cryptography and Computer Privacy

Horst Feistel (1915 - 1990)

6. SUC AS NEW FEISTEL-LIKE CIPHER DESIGN

In this chapter, a unique unconventional cipher design is presented. The design utilizes the proposed implementation strategy inspired from the fabric resources of the target Microsemi FPGA. In particular, the proposed cipher structure deploys FPGA specific available 18 x 18 bits multipliers and 4 to 1 LUTs in the design of the self-generated SUCs. On the other hand, the proposed cipher is designed based on a Feistel core permutation. Two new classes of the Feistel-Like cipher are presented and investigated. The cardinality of each new class is greater than 2^{500} ciphers. Furthermore, the necessary condition for such ciphers to be as PRF is fulfilled, which makes the created SUCs resistant to state-of-the-art modeling attacks or any type of distinguishing attacks.

The main advantage of this cipher structure is that the same core function (Feistel-Like permutation) is utilized for both encryption and decryption operations, differing only in using the keys in reverse orders. The involutive structure makes the designed ciphers consume the same hardware resources for both encryption and decryption operations. The proposed approaches show that the resulting cipher-classes exhibit relatively very low complexity in hardware implementation as well as a high level of security against several attacks. It is worth mentioning that the contents of this chapter have been published consecutively in [80], [99], [100], and [29].

6.1. Introduction

In the mid-seventies the National Bureau of Standards (NBS) announced a cryptographic algorithm for data-protection so-called DES [21]. The structure of DES deploys a Feistel permutation as a core function. Such a permutation was designed by Horst Feistel in the early seventies. DES is defined as a block cipher with 64 bits input size and 56 bits key size. On the other hand, several block ciphers can be perceived as a Feistel cipher [101] such as Camellia [102], LBlock [103], Piccolo [104], MIBS [105], SIMON and SPECK [106], etc. The common component between all these ciphers and DES is the Feistel permutation.

In [107], Diffie and Hellman theoretically presented a machine that exhausts all 2^{56} keys at a rate 10^{12} keys per second. Such a machine could recover the DES secret key in one day. After that several studies were published to reduce the time complexity of the exhaustive search. The first attempt in this regard was determined by Wiener in 1993. The results showed that the exhaustive search was dramatically reduced to 3.5 expected hours [108]. The second attempt in this regard was presented in [109] by Biham and Shamir who developed a differential cryptoanalysis method (DC). The results showed that DC recovers DES-secret key with up to 8 rounds in a few minutes and 15 rounds faster than an exhaustive search [109]. Even when the DES was designed to resist DC, the full 16 rounds of DES are breakable by 2^{47} chosen plaintexts [110]. In [111], the linear cryptoanalysis method (LC) was firstly proposed for breaking DES. This attack requires 2^{21} known-plaintexts to break 8-round DES and 2^{47} known-plaintexts to break 16-round DES.

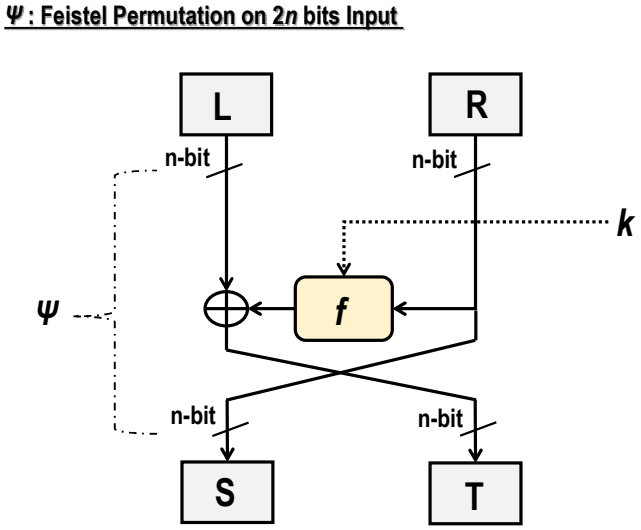


Figure 6-1. Sketch of Feistel Permutation.

Figure 6-1 shows the Feistel permutation on $2n$ bits input block size that divides into two n subblocks (L, R). The keyed function f operates on n bit right part R. The Feistel mapping is then,

$$(S,T)=\psi(f)(L,R)=(R,L\oplus f(R)) \tag{6-1}$$

In [112], Luby and Rackoff presented the construction of a super-pseudorandom

permutation (SPRP) (a secure block cipher) having four rounds of Feistel permutations with four independent different round functions. This work was a breakthrough and one of ground-breaking papers in cryptography. Since that time numerous studies were investigated and published regarding the most practical and realizable methods to ensure the reliability and high-security level of a block cipher based on a Feistel permutation [113]. In [114], Maurer presented a strongly simplified treatment of Luby-Rackoff results and generalized them. The proposed Luby-Rackoff SPRP can be described as follows:

$$(V,W)=\psi(f_1,f_2,f_3,f_4)(L,R) \tag{6-2}$$

Where,

$$\left.\begin{aligned} S &= L\oplus f_1(R)\\ T &= R\oplus f_2(S)\\ V &= S\oplus f_3(T)\\ W &= T\oplus f_4(V) \end{aligned}\right\} \tag{6-3}$$

And f_1, f_2, f_3, f_4 are independent pseudorandom functions.

The Luby-Rackoff formal model of a secure cipher is presented based on distinguishing attacks, where, the adversary tries to distinguish between the targeted Luby-Rackoff block cipher and a random permutation on the same input space. In [112], Luby and Rackoff proved that three independent pseudorandom keyed functions for Luby and Rackoff cipher are a PRP, whereas four independent pseudorandom keyed functions are a SPRP.

Theorem 6.1 (Luby-Rackoff [112] [115]**):** Let f_1, f_2, f_3, f_4 be independent pseudorandom keyed functions from F_n:, where, $|F_n| = 2^{n\cdot 2^n}$. Let B_1 be the family of permutations on $\{0,1\}^{2n}$ consisting of permutations such as $\psi(f_1, f_2, f_3)$. Then,

(6-4)

Let B_2 be the family of permutations on $\{0,1\}^{2n}$ consisting of permutations such as $\psi(f_1, f_2, f_3, f_4)$. Then,

$$Avd_{PRP}^{\psi(f_1,f_2,f_3,f_4)}(q,t)\le\binom{q}{2}\left(2^{-n+1}+2^{-2n}\right) \tag{6-5}$$

In [113], Patarin investigated the optimal number of plaintext-ciphertext pairs to distinguish a Luby-Rackoff cipher from a truly random permutation [116] [117]. The results showed that 7 rounds or more are secure against all CPA [118], and 10 rounds or more are secure against all CPA and CCA [117].

Several variants of the original Luby-Rackoff cipher $Avd_{PRP}^{\psi(f_1,f_2,f_3)}(q,t)\le\binom{q}{2}\left(2^{-n+1}+2^{-2n}\right)$ were investigated and presented. The first approach concentrated on minimizing the number of pseudorandom functions of Luby-Rackoff cipher [119]. For instance, Ohnishi's constructions $\psi(g, f, f)$ and $\psi(f, f, g)$ were proved as PRPs, where g, f are two independent PRFs [120]. In [121], Pieprzyk proved that for applying a PRF such as f internally at least

five times $\psi(f,f,f,f^2)$, the resulting cipher design is a PRP. Another approach was launched as new PRP such as $\psi(h_1, f_1, f_2, h_2)$ by using two PRFs f_1, f_2 together with two universal hash functions such as h_1, h_2 [122].

All previous constructions involve XOR operation. In [110], Biham and Shamir replaced some of the XOR operations in DES by addition mod 2^n, the resulting cipher becomes more resistant against DC. On the other hand, replacing the XOR operation of DES by * operation defined as a Latin Square can be perceived as a general approach towards non-XOR constructions of the Luby-Rackoff cipher [123]. Similarly, a new non-XOR construction of Luby-Rackoff cipher as $\psi(h, f, f, h)$ was presented in [124], where, f is a PRF, and h is a universal hash function. The resulting cipher structure uses addition mod 2^n instead of XOR operation. This work is motivated by the following fact: "*let $X=2^n-1$, $Y= 1$ be the integer representation of two n-bit blocks. In this case $X+Y$ mod 2^n is equal to zero, which means all bits in X are affected but in case of $X \bigoplus Y$, the answer is equal to n-1 ones and a zero in the last significant bit, and only the last significant bit is affected* [115]". The results showed the same level of security between XOR-based Luby-Rackoff cipher and the non-XOR constructions [115].

Following these works, the next section encompasses a new variant to the Luby-Rackoff cipher. The proposed Luby-Rackoff cipher can be classified as a non-XOR construction.

6.2. Proposed Feistel-Like Cipher Design

A new design of Luby-Rackoff cipher is presented below by replacing the XOR operation with a new reliable self-inverse mapping. The new mapping is designed based on MAACs over 2^n in SmartFusion®2 FPGA. As mentioned earlier, the key-idea for the MAACs is freely available as unused hardware components in SmartFusion®2 FPGA. Therefore, the new cipher design is reanimating the unused arithmetic components for creating a good cipher. The resulting new cipher-class is also usable for self-creating SUCs [80].

6.2.1. New Latin Square as Involution-Mappings

Let Π^2 denote the set of all polynomials $P : \mathbb{Z}_{2^n} \times \mathbb{Z}_{2^n} \to \mathbb{Z}_{2^n}$ of two variables of total degree 1 of form:

$$P(L,R) = aL \overset{\oplus}{+} bR \qquad (6\text{-}6)$$

Where, $a, b \in \mathbb{Z}_{2^n}$. In this case, any polynomial P from Π^2 is defined as a mapping having two inputs such as (L,R) and one output P(L,R) in $\mathbb{Z}_{2^n}$, where, $|L| = |R| = n$.

Definition 6.1 [125]**:** The polynomial P in two variables L and R from Π^2 defined over $\mathbb{Z}_{2^n}$ is considered as a Latin square, if both functions $P(L, C)$ and $P(C, R)$ are permutations of $\mathbb{Z}_{2^n}$, for any $C \in \mathbb{Z}_{2^n}$.

In [126], Klimov proved that "a polynomial of the form $P(x) = a_0 \overset{\oplus}{+} a_1 x \overset{\oplus}{+} \cdots \overset{\oplus}{+} a_d x^d$ is a permutation polynomial modulo 2^n: n >2 if and only if a_1 is odd, (a_2+a_4+…) is even, and (a_3+a_5+…) is even" (See theorem 7.5 chapter 7). Accordingly, the following lemma

determines the main requirements of $P(L,R) = aL \overset{\oplus}{+} bR$ to be a Latin square over $\mathbb{Z}_{2^n}$.

Lemma 6.2 [29]**:** Let $n>2$ and $P(L,R) = aL \overset{\oplus}{+} bR$ be a polynomial in two variables (L,R) over $\mathbb{Z}_{2^n}$. Then, P is a Latin square, if a and b are odd numbers.

Proof:

Without loss of generality, let's prove $P(L,R)$ of form $P(L, \mathrm{R})=aL+bR$ is a Latin square, other fashions can be proved in a similar way.

According to Klimov's theorem (see theorem 7.5), $P(L,C)=aL+bC$, and $P(C, R)=aC+bR$ are permutations over $\mathbb{Z}_{2^n}$., if a and b are odd numbers. This implies that $P(L, R)= a.L+b.R$ is a Latin Square based on definition.6.1. □

Definition 6.3: The polynomial P in two variables L and R from Π^2 is considered a self-inverse mapping with respect to L over $\mathbb{Z}_{2^n}$, if it holds:

$$P(P(L, R), R) \bmod 2^n = L \tag{6-7}$$

for every L and R.

The following theorem determines the main requirements of a Latin square $P(L, R)$ to be a self-inverse mapping with respect to L over $\mathbb{Z}_{2^n}$.

Theorem 6.4: Let $n>1$ and $P(L,R) = aL \overset{\oplus}{+} bR$ be the defined Latin square over $\mathbb{Z}_{2^n}$. Then, P is a self-inverse mapping with respect to L if $a = \underbrace{1 \cdots 1}_{n} = 2^n - 1$.

Proof:

Note that if $a=2^n-1$, then $a^2=(2^n-1)^2=2^{2n} -2^{n+1} +1$, so that $a^2 \bmod 2^n=1$. Now, let,

$$P(P(L, R), R)= a.(a.L+b.R) +b.R$$

And,

$$P(P(L, R), R)= a^2L+b(a+1)R$$

Now,

$$P(P(L, R), R)= a^2L+b(2^n-1+1)R$$

Yielding,

$$P(P(L, R), R)= a^2L+2^nbR$$

Applying mod 2^n results with, $P(P(L, R), R) \bmod 2^n=L$.

(the remaining other cases for – and $\oplus$ can be proved in a similar way). □

Let Π_i denote special classes of self-inverse mapping with respect to L from Π^2, for i=1,2,3, as follows,

$$\Pi_1 : P(L,R) = aL + bR;\ \Pi_2 : P(L,R) = aL - bR, \text{ and } \quad \Pi_3 : P(L,R) = aL \oplus bR \tag{6-8}$$

The following corollary determines the cardinality of the classes Π_i ; i=1,2,3.

Corollary 6.5: For n>3, the cardinality of the class Π_i of all possible $P(L,R)=aL \overset{\oplus}{+} bR$ over $\mathbb{Z}_{2^n}$ is:

$$\text{Card}(\Pi_i)=2^{n-1} \tag{6-9}$$

For i=1,2,3.

Proof:

For n even, and from theorem 6.4, the following is true:

$$a=\underbrace{1\cdots 1}_{n} \Rightarrow |a|=1$$

And,

$$b \text{ is odd} \Rightarrow |b|=2^{n-1}$$

That implies, $|\Pi_i|=|a|\cdot|b|=1\cdot 2^{n-1}=2^{n-1}$. □

The resulting classes of self-inverse Latin squares Π_i: $P(L,R)$ define so-called π_i-mappings as delineated in Figure 6-2, where, $\pi_i(L,R)=(aL \overset{\oplus}{+} bR) \bmod 2^n$: i=1,2,3.

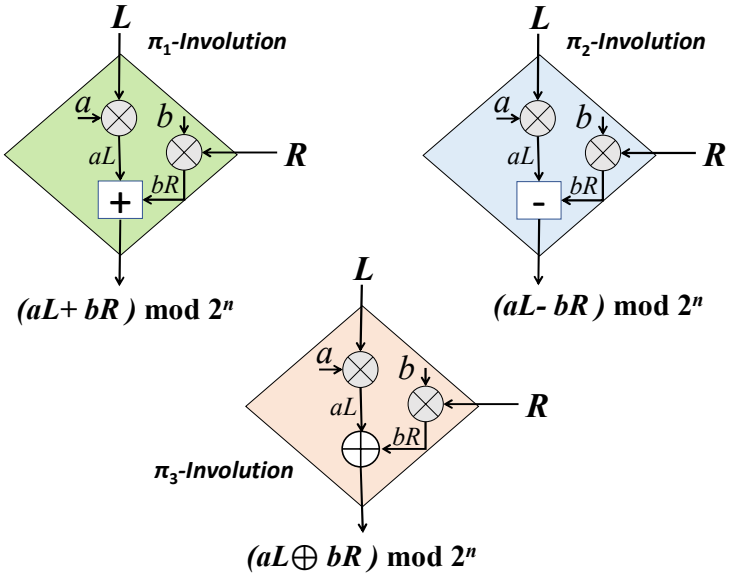


Figure 6-2. New π_i-Mappings used as an Arithmetic Operation [29].

Let's prove the π_1-mapping $\pi_1(L,R)=(aL+bR) \bmod 2^n$ is an involution with respect to L and other cases of – and $\oplus$ can be proved in a similar way.

For any R,

$$\pi_1(\pi_1(L,R),R)=\pi_1((aL+bR) \bmod 2^n, R)$$

And,

$$\pi_1(\pi_1(L,R),R)=(a(aL+bR)+bR) \bmod 2^n=(a^2L+b(a+1)R) \bmod 2^n$$

So that,

$$\pi_1(\pi_1(L,R),R) = L$$

Which means, the π_1-mapping $\pi_i(L,R) = (aL + bR) \bmod 2^n$ is an involution with respect to L.

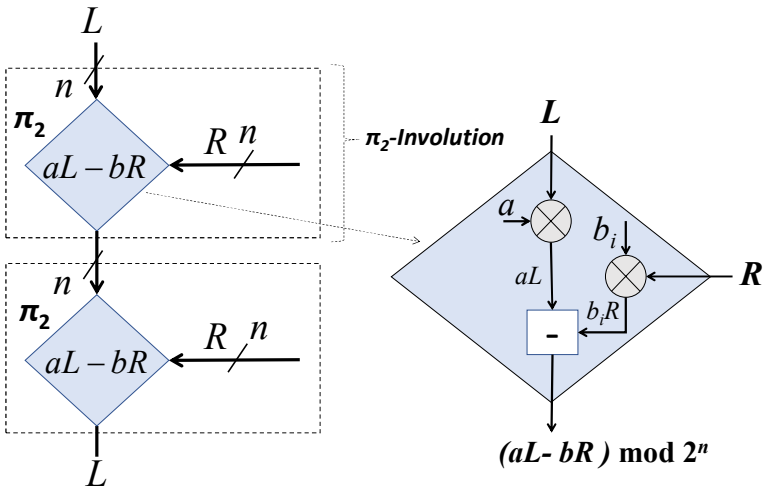


Figure 6-3. The New π-Mapping as an Involution deploying Arithmetic Operations. Adapted from [29].

Note that it is very simple to show that the π_i-mappings are an involution for any R as illustrated in Figure 6-3. The statistical properties of the multiplication and addition ensure that all the bits of the input will be affected. Moreover, such construction is very efficient in its implementation in case that SmartFusion®2 FPGA provides such specific MAACs for π_i-mappings as unused components.

Now, replacing the XOR-operation in Luby-Rackoff cipher with π_i-mappings results with a new mapping defined as follows:

$$\zeta(f)(L,R) = (aL \overset{\oplus}{+} bf(R), R) \tag{6-10}$$

Figure 6-4 shows the core mapping ζ presented in (6-10) for π_1-mapping of the proposed Feistel-like cipher. The round's input data is $2n$ bits that splits into two branches of n-bits (L: left and R: right). Then, ζ -Involution is applied on both branches (L, R), where, the inner function f is applied only on R. By applying $\zeta(f)$ two consecutive times on any L, R, it is very simple to prove that $\zeta(f)$ is an involution for any f.

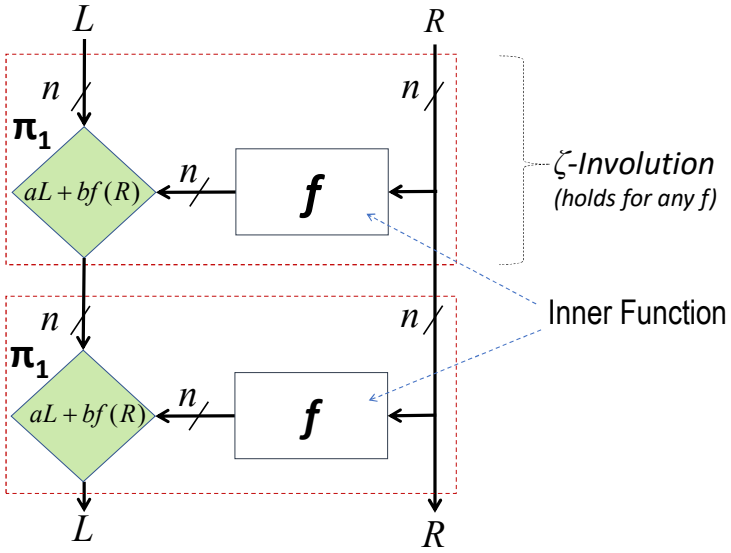


Figure 6-4. The New ζ-Involution as XOR Replacement [80].

The round function η of the proposed Feistel-Like cipher can be constructed by using ζ-Involution followed by a swap mapping as depicted in Figure 6-5. The t-rounds of η are using the same 2-involutions each time with a randomly chosen $(n-1)$ bits of b_i. Note that the total number σ_1 of all possible SUCs as Feistel- Like ciphers with t rounds is generally related to the total number μ of all possible inner functions f.

$$\sigma_1 = \max_{\mu}\left\{\left(2^{(n-1)}\right)^t \cdot \mu\right\} = 2^{t(n-1)} \cdot 2^{n2^n} = 2^{t\cdot(n-1)+n2^n} \tag{6-11}$$

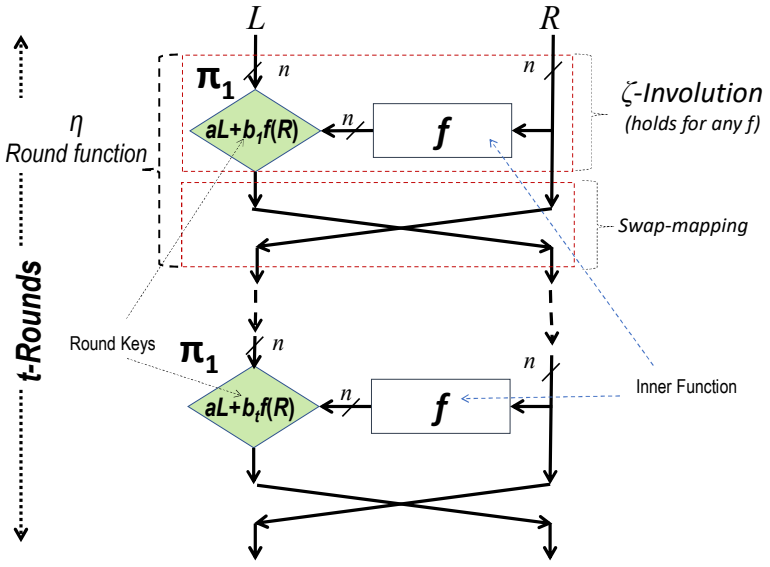


Figure 6-5. The proposed Feistel-like cipher structure [29] and [80].

The main advantage of the proposed cipher structure is that the same round function is utilized for both encryption and decryption operations, differing only in using the keys in reverse orders. In the following two rounds of the proposed Feistel-Like cipher are presented as an example for encryption and decryption modes.

Encryption Mode:

$$\eta(f,f)(L,R)=\eta(f)(R,aL+b_1f(R))=(aL+b_1f(R),aR+b_2f(aL+b_1f(R)))=(S,T)$$

Decryption Mode:

$$\eta^{-1}(f,f)(S,T)=(D\circ\bar{\eta}(f,f)\circ D)(S,T)$$

Where, D is the swap-mapping and $\bar{\eta}(f,f)$ is the function $\eta(f,f)$ with the keys (b_1 and b_2) in reverse orders. This implies:

$$(D\circ\bar{\eta}(f,f)\circ D)(S,T)=(D\circ\bar{\eta}(f,f))(aR+ab_2f(aL+b_1f(R)),aL+b_1f(R))$$

And,

$$(D\circ\bar{\eta}(f,f)\circ D)(S,T)=(D\circ\bar{\eta}(f))(aL+b_1f(R),a^2R+ab_2f(aL+b_1f(R))+b_2f(aL+b_1f(R)))$$

By applying mod 2^n:

$$(D\circ\bar{\eta}(f,f)\circ D)(S,T)=(D\circ\bar{\eta}(f))(aL+b_1f(R),R)$$

And,

$$(D\circ\bar{\eta}(f,f)\circ D)(S,T)=(D)(R,a^2L+ab_1f(R)+b_1f(R))$$

By applying mod 2^n:

$$(D\circ\bar{\eta}(f,f)\circ D)(S,T)=(D)(R,L)$$

So that,

$$\eta^{-1}(f,f)(S,T)=(D\circ\bar{\eta}(f,f)\circ D)(S,T)=(L,R)$$

The previous procedures can be repeated for t rounds for both encryption and decryption modes.

6.2.2. Distinguishing Attack on the Proposed Feistel-Like Cipher

The evaluation of distinguishing attack on the proposed Feistel-like cipher is carried by deploying the core mapping η as a mapping in different ciphering configurations. These structures can be developed based on distinguishing attack scenarios such as *Distinguishing Experiment-2* in chapter 2.

The one round Feistel-like cipher is described as:

$$(S_i, T_i)=\eta(f)\,(L_i, R_i)=(P(L_i,f(R_i)), R_i)=(aL_i+b_1f(R_i), R_i) \quad (6\text{-}12)$$

For 1 round:

$$\left.\begin{aligned} S_i &= R_i \\ T_i &= aL_i+b_1f(R_i) \end{aligned}\right\} \quad (6\text{-}13)$$

The adversary can just test if $S_i = R_i$ for every i. This will happen with 100% probability after one query. Therefore, one round of the proposed Feistel-like cipher is not a PRP.

For 2 rounds: The proposed Feistel-like cipher $\eta(f,f)$ (L_i, R_i) can be described as:

$$\left.\begin{aligned} S_i &= aL_i + b_1 f(R_i) \\ T_i &= aR_i + b_2 f(S_i) \end{aligned}\right\} \tag{6-14}$$

In case of $b_1=b_2$, the adversary chooses two pairs (L_1, R_1) and (L_2, R_2), where, $R_1=R_2$ and $L_1 \neq L_2$. Then, the adversary can just test if S_1- S_2= $a(L_1-L_2)$. This will happen with 100% probability after four queries. Therefore, the proposed Feistel-like cipher with two rounds is not a PRP if $b_1=b_2$.

For 3 rounds: The proposed Feistel-like cipher η (f,f,f) (L_i, R_i) can be described as:

$$\left.\begin{aligned} X_i &= aL_i + b_1 f(R_i) \\ S_i &= aR_i + b_2 f(X_i) \\ T_i &= aX_i + b_3 f(S_i) \end{aligned}\right\} \tag{6-15}$$

In case of $b_1=b_2=b_3$, the adversary performs the following steps:

- Choose (L_1, R_1)=(0,0) as query for η (f,f,f) resulting with (S_1, T_1) .
- Choose (L_2, R_2)=(0,S_1) as query for η (f,f,f) resulting with (S_2, T_2).
- Choose (L_3, R_3)=(T_1 -aT_2, S_2) as query for η (f,f,f) resulting with (S_3, T_3).

Then, the adversary can just test if S_3= aS_2 + S_1. This will happen with 100% probability after at most $O(2^{n+1})$=O(2^{n+1}+2^n) queries. Therefore, the proposed Feistel-like cipher with three rounds is not a PRP if $b_1=b_2=b_3$.

We can now rephrase the previous results as follows:

None of η (f), η (f,f) and η (f,f,f) are PRPs where $b_1=b_2=b_3$. However, Luby and Rackoff proved in their seminal paper [112] that Luby-Rackoff cipher $\psi(g, f, h)$ is a PRP using three independent PRFs (g, f, h).

To minimize the number of PRFs of Luby-Rackoff cipher [119], for instance, $\psi(g,f,f)$ was proved as an optimal PRP in [120]. Following this work, it is too easy to show that by using Maurer's simplified treatment suggested in [114], the proposed Feistel-like cipher η(g,f, h) is a PRP and $\eta(g, f, f)$ as well (See Appendix B).Therefore, such approach doesn't lead to minimize the number of used PRFs in the proposed Feistel-like cipher.

In the following, a new design of a PRP is constructed by using single PRF. The proposed Feistel-like cipher η_3 (f) with $b_1 \neq b_2 \neq b_3$ is described as:

$$\left.\begin{aligned} X_i &= aL_i + b_j f(R_i) \\ S_i &= aR_i + b_k f(X_i) \\ T_i &= aX_i + b_l f(S_i) \end{aligned}\right\} \tag{6-16}$$

Where, $f \xleftarrow{U} F$ and $j \neq k \neq l$. Here, η_t (f) is $\eta\left(\underbrace{f,\ldots,f}_{t}\right)$ with t different random values of b_i; i=1,…,t.

To prove that the proposed η_3 (f) deploying a single PRF $f \xleftarrow{U} F_n$ is indistinguishable from a truly random permutation, the Distinguishing Experiment-2 should be applied on $\eta_3(f)$. Here, the proof of $\eta_3(f)$ is a PRP, hence, an immediate consequence of the following Lemma.

Lemma 6.6: For every function $G:\left(\{0,1\}^{2n}\right)^q \to \{0,1\}$ and for any q pairs (L_i, R_i) from $\{0,1\}^n$ x$\{0,1\}^n$, i=1,…q.

$$\left|\Pr\left[G\left(\eta_3(f)(L_1,R_1),\cdots,\eta_3(f)(L_q,R_q)\right)=1 : f \xleftarrow{U} F_n\right]-\Pr_G\right| \le \frac{q^2}{2^n} \tag{6-17}$$

Where, $f \xleftarrow{U} F$ and $\Pr_G$ defined as:

$$\Pr_G = \frac{\#\{(x_1,\cdots,x_q)\in\left(\{0,1\}^{2n}\right)^q : G(x_1,\cdots,x_q)=1\}}{2^{2nq}} \tag{6-18}$$

Proof:

Assume without loss of generality that the q pairs (L_i, R_i) are distinct. According to (6-16), the outputs of the first, second, and third round are (R_i, X_i), (X_i, S_i), and (S_i, T_i), respectively. Let A_X be the event that $\{X_i\}_{i=1}^q$ are distinct and let A_S be the event that $\{S_i\}_{i=1}^q$ are distinct. Then, $A_X \cap A_S$ is the event that $\{X_i\}_{i=1}^q$ and $\{S_i\}_{i=1}^q$ are distinct.

Now, if the event A_X occurs, then the values $S_i=aR_i+b_kf(X_i)$ are random for i=1,…q, where, $b_kf(X_i)$ is a multiplication of two random values. On the other hand, $f_l \xleftarrow{U} F$ and $b_l \xleftarrow{U} \{0,1\}^n$, therefore, if the event A_S occurs, then the values $T_i=aX_i+b_lf(S_i)$ are random for i=1,…q. In this case, $\eta_3(f)$ behaves precisely like a randomly chosen function from the set of all possible functions F_{2n}, and the probability of distinguishing between $\eta_3(f)$ and a random function from F_{2n} is:

$$\left|\Pr\left[G\left(\eta_3(f)(L_1,R_1),\cdots,\eta_3(f)(L_q,R_q)\right)=1 : f \xleftarrow{U} F_n\right]-\Pr_G\right| \le 1-\Pr[A_X \cap A_S] \tag{6-19}$$

And,

$$1-\Pr[A_X \cap A_S] = \Pr[\overline{A_X \cap A_S}] = \Pr[\overline{A_X} \cup \overline{A_S}] \le \Pr[\overline{A_X}]+\Pr[\overline{A_S}] \tag{6-20}$$

Where, $\overline{A_X}$ ($\overline{A_S}$) in the complementary event of A_X (A_X) occurring when $\{X_i\}_{i=1}^q$ ($\{S_i\}_{i=1}^q$) are not distinct, respectively.

For $i \neq j$, and according to the main assumption: the q pairs (L_i, R_i) are distinct.

$$\Pr[\overline{A_X}] = \binom{q}{2}\sum_{1\le i<j\le q}\Pr[X_i=X_j], \text{and } \Pr[\overline{A_S}] = \binom{q}{2}\sum_{1\le i<j\le q}\Pr[S_i=S_j] \tag{6-21}$$

Where, $\binom{q}{2}$ is the number of choosing 2 equal values [X_i=X_j] ([S_i=S_j]) out of q from $\overline{A_X}$ ($\overline{A_S}$), respectively. On the other hand, the q pairs (L_i, R_i) are distinct by assumption, Pr[X_i=X_j] and Pr[S_i=S_j] are computed as,

$$\Pr[X_i = X_j] = \begin{cases} 2^{-n}; & R_i \neq R_j \\ 0 & ; R_i = R_j \end{cases} \text{,and} \quad \Pr[S_i = S_j] = \begin{cases} 2^{-n}; & X_i \neq X_j \\ 0 & ; X_i = X_j \end{cases} \tag{6-22}$$

From (6-21) and (6-22),

$$\Pr[\overline{A_X}] = \binom{q}{2} \cdot 2^{-n} \text{,and } \Pr[\overline{A_S}] = \binom{q}{2} \cdot 2^{-n} \tag{6-23}$$

Substituting (6-23) by (6-20),

$$1 - \Pr[A_X \cap A_S] \leq 2 \cdot \binom{q}{2} \cdot 2^{-n} = \frac{q(q-1)}{2^n} < \frac{q^2}{2^n} \tag{6-24}$$

Call (6-19), we obtain,

$$\left| \Pr\left[G\left(\eta_3(f)(L_1, R_1), \cdots, \eta_3(f)(L_q, R_q)\right) = 1 : f \xleftarrow{U} F_n \right] - \Pr_G \right| \leq \frac{q^2}{2^n} \qquad \square$$

Distinguishing Experiment-2 for $\eta_3(f)$:

Step 1: For the proposed Feistel-like cipher $\eta_3(f)$, consider an adversary (distinguisher) Ψ that interacts with a challenger C who works as follows:

- C randomly chooses one bit $b \xleftarrow{U} \{0,1\}$.
- C returns $P \xleftarrow{U} B_{2n}$ if b=1 to Ψ, otherwise returns $P \xleftarrow{D} \eta_3(f)$, where $f \xleftarrow{U} F$.

Within time t.

Step 2: The adversary Ψ submits to challenger C a polynomial number of queries (q) such as (L_i, R_i) from $\{0,1\}^n$ x$\{0,1\}^n$, i=1,…q.

Step 3: The adversary terminates the experiment by returning b'.

According to lemma 6.6, the advantage of Ψ to distinguish between $\eta_3(f)$ and a random function is:

$$Adv_{PRF}^{\eta_3(f)}(\Psi) \leq \frac{q^2}{2^n} \tag{6-25}$$

Now, the PRF Switching Lemma (lemma 2.1) stated that,

$$Adv_{PRP}^{\eta_3(f)}(\Psi) < Adv_{PRF}^{\eta_3(f)}(\Psi) + q^2 / 2^{n+1} \tag{6-26}$$

Observe that if q is not of order 2^n, it is not possible to distinguish $\eta_3(f)$ from a random permutation with a high probability.

Lemma 6.7: For $q<2^n$, the advantage of Ψ to distinguish between a random permutation $P \xleftarrow{U} B_{2n}$ and the proposed cipher $P \xleftarrow{D} \eta_3(f)$, where $f \xleftarrow{U} F$, is

$$Adv_{PRP}^{\eta_3(f)}(\Psi) < 3q^2 / 2^{n+1} \tag{6-27}$$

The previous lemma concludes the main result in this section, where, the proposed Feistel-Like ciphers attain the same security bound of Luby-Rackoff cipher.

6.3. Cipher Design Building Elements: Mappings and Operators

In this section, the necessary design building elements for the inner function *f* with good security properties are presented. The target is to design a huge class of cryptographically significant mappings with low-cost implementation as a proposal for the inner functions *f* of the proposed cipher $\eta_t(f)$.

6.3.1. Golden 4-bit S-Boxes as Primitive Building Elements

Let *x* denote a vector $x=(x_{n-1},\cdots,x_0)$ in $\mathbb{F}_2^n$, the inner product of two vectors such as *x* and *y* is defined as follows,

$$< x, y >= \sum_{i=0}^{n-1} x_i y_i \tag{6-28}$$

Let *h*(.) denote a binary Boolean function from $\mathbb{F}_2^n$ to $\mathbb{F}_2$. Whereas, a Boolean function *S*(.) from $\mathbb{F}_2^n$ to $\mathbb{F}_2^n$ is defined as a combination of *n* binary Boolean functions h_i(.). Furthermore, for a given Boolean function *S* from $\mathbb{F}_2^n$ to $\mathbb{F}_2^n$, the Fourier coefficient at point (*a*,*b*) is defined as,

$$W_{a,b}^S = \sum_{x\in\{0,1\}^n} (-1)^{a\cdot x + b\cdot S(x)} \tag{6-29}$$

And the linearity of a Boolean function *S* is defined as follows [127],

$$Lin(S) = \max_{a,b\in\mathbb{F}_2^n, b\neq 0} \left| W_{a,b}^S \right| \tag{6-30}$$

The linearity of *S* represents a measure for the resistance against LC. If *Lin*(*S*) is very small, then *S* is secure against LC. Furthermore, the linear probability bias ε is estimated as,

$$\varepsilon \leq \left| \frac{Lin(S)}{2^{n+1}} \right| \tag{6-31}$$

To analyze the resistance against DC, let $|\Delta_{S,a}^{-1}(b)|$ be the number of message pair $(x, x+a)$, with the output difference *b* of a Boolean function *S*. The resistance of a Boolean function *S* against DC is defined as follows,

$$Diff(S) = \max_{a,b\in\mathbb{F}_2^n, a\neq 0} |\Delta_{S,a}^{-1}(b)| \tag{6-32}$$

Where,

$$\Delta_{S,a} : \mathbb{F}_2^n \to \mathbb{F}_2^n \qquad x \to S(x) + S(x+a) \tag{6-33}$$

In the following, a Boolean function from $\mathbb{F}_2^4$ to $\mathbb{F}_2^4$ is called S-box and denoted by $S(.)$. In [127], Leander *et al.* classified all optimal 4-bit Boolean functions (S-Boxes) which satisfy: S is a bijection function, $Lin(S) = 8$ and $Diff(S) = 4$. The results showed that there are 16 different classes of optimal S-boxes, i.e., they are resistant to LC and DC.

Definition 6.3 [127]**:** We say that two S-Boxes S and S' are equivalent and belong to the same linear equivalence class, if there are two invertible 4x4 matrices A,B and $a,b \in \mathbb{F}_2^4$ such that,

$$S'(x)=B(S(A(x)+a))+b \tag{6-34}$$

For every $x \in \mathbb{F}_2^4$.

In [128], Saarinen showed that four of 16 optimal classes can affinely transform the resistance properties against LC and DC to the all members of classes. These optimal 4-bit S-Boxes are called golden S-Boxes (GSs) that are defined based on a new equivalence relation using two bit-permutation matrices P_i, P_k, two values $a,b \in \mathbb{F}_2^4$, and XOR-operation as follows,

$$[S(x)]_{1\times4} = GS_k[([x]_{1\times4} \oplus [a]_{1\times4}) \cdot [P_i]_{4\times4}] \cdot [P_j]_{4\times4} \oplus [b]_{1\times4} \tag{6-35}$$

Where, GS_j is a golden S-Box for j=0,1,2,3, (See Table. II) and $x \in \mathbb{F}_2^4$. The cardinality of the class of all possible GSs is then,

$$4.(2^4)^2.\,(4!)^2 = 2^{19.1} \quad \text{different GSs} \tag{6-36}$$

Where, 4 is the number of GS classes GS_j, $(2^4)^2$ is the number of all possible values $a,b \in \mathbb{F}_2^4$, and $(4!)^2$ is the number of all possible bit-permutation matrices P_i, P_j.

Table II. Four Golden S-Box Seeds [128]

GS Seed-Classes	**4-bit input combinations 0123456789ABCDEF**	**DC** p	**LC** ε
GS₀: 4-bit outputs	035869C7DAE41FB2	¼	¼
GS₁: 4-bit outputs	03586CB79EADF214	¼	¼
GS₂: 4-bit outputs	03586AF4ED9217CB	¼	¼
GS₃: 4-bit outputs	03586CB7A49EF12D	¼	¼

ε: linear probability bias, p: differential characteristics probability

Table II shows four GS-Seeds which can be deployed to generate four GS-classes by using (6-35). All members (GSs) in these classes satisfy the ideal security properties.

However, a modification of (6-35) allows to create a subclass simply by more practical and efficient mapping implementation as follows [99],

$$[S(x)]_{1\times4} = GS_k[[x]_{1\times4}] \cdot [P_i]_{4\times4} \oplus [a]_{1\times4} \tag{6-37}$$

Figure 6-6 shows the hardware sketch of two GS-generators according to (6-35) and (6-37). The resulting S-Boxes are cryptographically equivalent. It is very important to notice that only 19584 members from each class have a (single) cycle structure property [128].

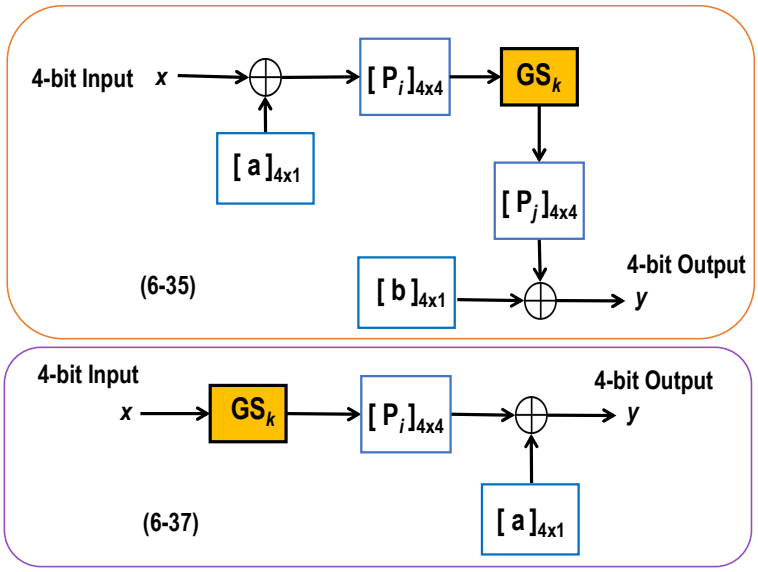


Figure 6-6. Hardware Sketch of the GS-Generators [29] , [99].

6.3.2. Bundle Permutations as Primitive Building Elements

A Bundle Permutation (BP) is a permutation that changes the positions of the bits bundles without changing the positions of the bits within a bundle [20]. Recently, BPs were used in modern cipher such as RC6 [129], and Khudra [98]. Moreover, Type-1, Type-2 and Type-3 of Generalized Feistel Networks (GFN) [130] deploy the same BP as a permutation layer [99].

However, the question remains how many iterations of BP the cipher needs to reach full diffusion. The cycle structure property of BP and its branch number are the most important factors to answer this question. In [131], the diffusion property of Type-2 GFN is improved by replacing the cyclic shift by an optimized BP. Such a permutation allows to reduce the number of rounds to attain sufficient security levels.

Definition 6.4: Let l be an even integer, (m, l)-BPs is a mapping over $\left(\{0,1\}^m\right)^l$ of l bundle's branches/subblocks with m-bit size of every subblock defined as:

$$\left(Y_1, Y_2, \ldots, Y_k\right) \rightarrow BP\left(Y_1, Y_2, \ldots, Y_k\right) \tag{6-38}$$

Therefore, BP is a shuffle of l subblocks [100].

In [99], an exhaustive search has been performed to test the (single) cycle structure preparty of all $(m, 4)$-BPs and $(m, 3)$-BPs (see Figure 6-7). The exhaustive search showed that 6 out of all 24 $(m, 4)$-BPs and 2 out of all 6 $(m,3)$-BPs have the cycle structure preparty (see table III and IV).

Table III. A Class for the Bundle Permutations with 3 Bundle's subblocks [99].

$(m,3)$-BP	**Y_0**	**Y_1**	**Y_2**
$(m,3)$-BP$_0$	Y_1	Y_2	Y_0
$(m,3)$-BP$_1$	Y_2	Y_0	Y_1

Note that BP with a cycle structure property minimizes the number of iterations the cipher needs to reach full diffusion.

Table IV. A Class for the Bundle Permutations with 4 Bundle's subblocks [99].

(*m*,4)-BP	Y_0	Y_1	Y_2	Y_3
(*m*,4)-BP$_0$	Y_1	Y_2	Y_3	Y_0
(*m*,4)-BP$_1$	Y_1	Y_3	Y_0	Y_2
(*m*,4)-BP$_2$	Y_2	Y_0	Y_3	Y_1
(*m*,4)-BP$_3$	Y_2	Y_3	Y_1	Y_0
(*m*,4)-BP$_4$	Y_3	Y_0	Y_1	Y_2
(*m*,4)-BP$_5$	Y_3	Y_2	Y_0	Y_1

Furthermore, every (*m*,4)-BP corresponds a binary matrix. For instance, (*m*,4)-BP$_0$ is equivalent to the following matrix;

$$A_0 = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{pmatrix} \Rightarrow \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} Y_0 \\ Y_1 \\ Y_2 \\ Y_3 \end{pmatrix} = \begin{pmatrix} Y_1 \\ Y_2 \\ Y_3 \\ Y_0 \end{pmatrix} \tag{6-39}$$

The corresponding matrices of all (*m*,4)-BPs are represented as:

$$\left.\begin{aligned} A_1 = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}; A_2 = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{pmatrix}; A_3 = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}; \\ A_4 = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}; A_5 = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}; \end{aligned}\right\} \tag{6-40}$$

Figure 6-7 shows two examples of the BPs (*m*, 4)-BP and (*m*, 3)-BP with *m*-bit size of every subblock. In this example, the presented (*m*, 4)-BP has a cycle structure preparty, that means it reaches full diffusion after 4 rounds but the presented (*m*, 3)-BP doesn't have a cycle structure preparty but it's an involution.

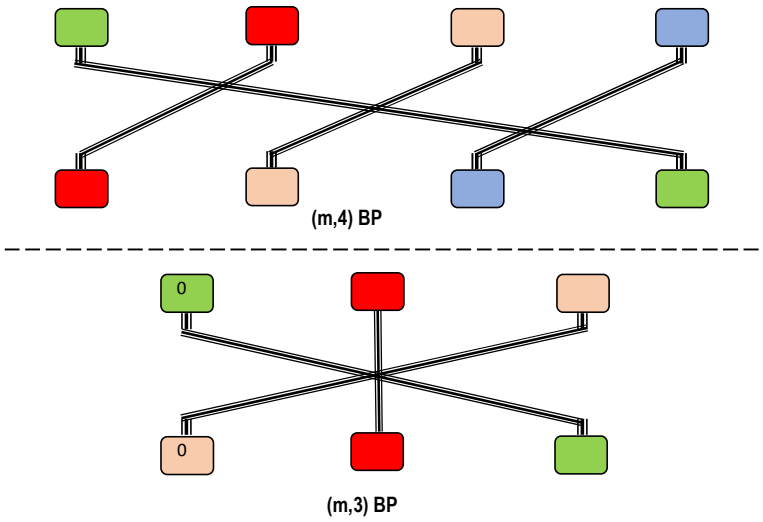


Figure 6-7. Two Examples of the Bundle Permutations [99].

6.4. FC_1: New Class of a Feistel-Like Cipher

In this section, a new class FC_1 of Feistel-like ciphers is proposed. The inner function of the Feistel-like cipher is designed based on a GFN type-2 [130] together with randomly chosen *GS*s and a BP stage.

Figure 6-8 shows three Types of GFN [130]. For instance, the input data size of GFN type-2 is $2n$ bits that splits into 4 bundles of $n/2$-bits. Two different functions F_1, F_2 from $\{0, 1\}^{n/2}$ to $\{0,1\}^{n/2}$ are applied on the bundles with even subscript as Y_0, Y_2. The GFN type-2 structure includes then 5-mappings namely: two XOR-operations together with two mappings F_1, F_2 followed by a BP. The r-rounds of such structure are using the same 5- mappings each time.

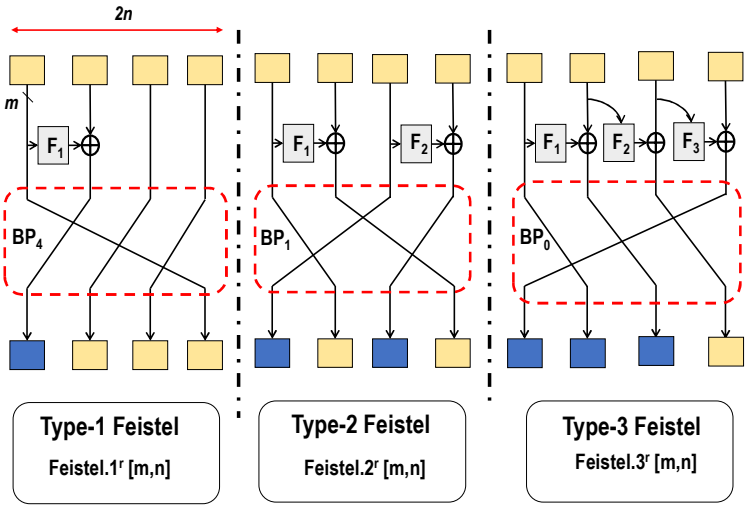


Figure 6-8. Three Different GFN Structures [99].

In [132], Hoang and Rogaway proved that the CCA- adversary has negligible advantage to distinguish GFN type-2 with $(m,4)$-BP_4 from a random permutation if it asks $q=2^{n(1-\varepsilon)/2}$

queries, where, $\varepsilon > 0$. Moreover, similar bounds were determined for GFN type-1 with (m,4)-BP_4 and GFN type-3 with (m,4)-BP_4 as well.

The proposed inner function f_1 is constructed by replacing XOR-operations in GFN type-2 with ζ-mapping resulting with a new mapping [80]. Figure 6-9 illustrates the round function of the Feistel-like cipher with the proposed inner function f_1. The resulting ciphers-class with the inner function f_1 is denoted by FC_1.

In Figure 6-9, the $2n$ bits cipher round splits into two n-bit subblocks (L, R). The ζ-mapping is applied on L and $f_1(R)$ [80], where, the inner function f_1 takes the n-bit R as an input. In particular, the inner function f_1 is performed by splitting R into four $n/4$-bit subblocks. Here, the involution ζ-mapping is used recursively again/repeatedly in $n/4$-bit size for the inner function f_1. Note that the number of the inner function rounds r should be decided based on both security and efficiency.

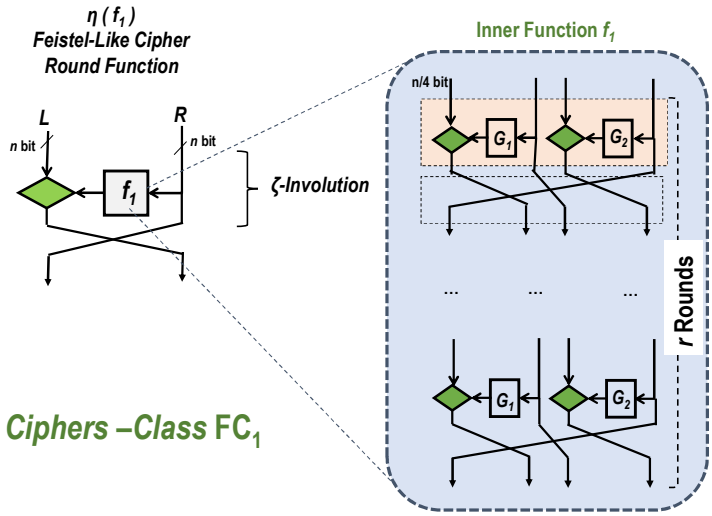


Figure 6-9. The Inner Function Structure f_1 for a Ciphers-Class FC_1. Adapted from [80].

Observe that the class FC_1 cardinality is related to the number of cipher rounds t, the number of the inner function rounds r, and the total number of all possible generated functions f_1. That implies,

$$|FC_1| = \left(2^{(n-1)}\right)^t \cdot \mu_1 = 2^{t(n-1)} \cdot 6 \cdot \left(2^{\frac{n}{4}\cdot 2^{\frac{n}{4}}}\right)^2 \cdot 2^{r\left(\frac{n}{4}-1\right)} \tag{6-41}$$

Where, μ_1 is the total number of all possible inner functions f_1. So that,

$$|FC_1| = 2^{t(n-1)} \cdot 6 \cdot 2^{\frac{n}{2}\cdot 2^{\frac{n}{4}}} \cdot 2^{r\left(\frac{n}{4}-1\right)} = 6 \cdot 2^{t(n-1)+\frac{n}{2}\cdot 2^{\frac{n}{4}}+\frac{r\cdot n}{4}-r} \approx 2^{t(n-1)+\frac{n}{2}\cdot 2^{\frac{n}{4}}+\frac{r\cdot n}{4}-r+2} \tag{6-42}$$

6.4.1. A New Design of the Inner Function

Assume that the mapping's input data of FC_1 in Figure 6-10 is $2n$=64 bits that divides into two 32-bit subblocks (L,R) in the first round. A 32-bit key determines b_i in $aL+b_i f_1(R)$ each

time. The $\eta(f)$ as $(R, aL+b_i\ (f_1(R))$ is applied on L and $f_1(R)$. The mapping f_1 takes the 32-bit R as input and splits it into four 8-bit subblocks R_i :i=1,2,3,4. Here f_1 consists of a nonlinear layer of two-horizontal-mappings G_1 and G_2 together with a randomly chosen (4,4)-BP as shown in Figure 6-10. The G_1 and G_2 structures utilizes four different GSs. The ζ-mapping is used recursively again in 8-bit size.

It is concluded that the total number $FC_{1,G1,G2}$ of all possible generated SUCs, as Feistel-Like ciphers with $2n$=64 bits input size and t=16 rounds, is computed as,

$$| FC_{1,G_1,G_2} |= 2^{t(n-1)} \cdot \mu_1 = 2^{16\cdot(32-1)} \cdot \mu_1 \qquad (6\text{-}43)$$

In Figure 6-10, the inner function f_1 deploys 4 GSs out of $2^{19.1}$ possible GSs for G_1 and G_2 and f_1 deploys one of the BPs from {(4,4)-BP_i }; i=1,2,3,4,5,6. Here the randomly chosen (8-1)=7 bits is utilized for b in ($aL+bR$). Therefore, the total number μ_1 of all possible inner functions f_1 is given as follows:

$$\mu_1 = 6\cdot\left(2^{19.1}\right)^4 \cdot 2^{(8-1)} = 2^{85.9} \qquad \text{different } f_1 \text{ inner functions} \qquad (6\text{-}44)$$

So that,

$$| FC_{1,G_1,G_2} |= 2^{16\cdot(32-1)} \cdot 2^{85.9} = 2^{581.9} \approx 2^{582} \qquad \text{different ciphers} \qquad (6\text{-}45)$$

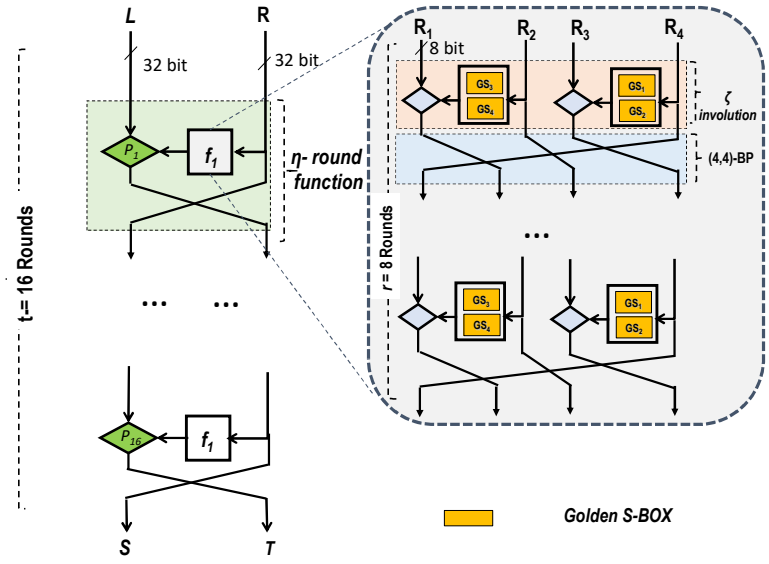


Figure 6-10. A 64-bit SUC as Feistel – Like Cipher FC_1 [80].

Furthermore, r=8 is the number of inner rounds decided based on both security and efficiency. From the implementation point of view, it is very efficient to choose the number of rounds as 2, 4,8,16,32, etc. On the other hands, such approach to design f_1 ensures a high level of security against DC and LC for the proposed cipher FC_1. The security level is analyzed based on the number of active GSs in differential and linear trails.

In the following the number of active GSs in differential and linear trails is defined, where, a differential trail is a sequence of input and output differences to the rounds, whereas, a linear trail is a sequence of round masks. Here, if c and x are two binary vectors such that $c^T.x$, we

call c the mask of x. Analyzing the highly likely differential and linear trials provide information about FC_1 security level.

Definition 6.5 [133]**:** In DC, an S-box is active in a differential trail if and only if its input difference is nonzero. In LC, an S-box is active in a linear trail if and only if its output mask is nonzero.

From (6-39) and (6-40), the (4,4)-BP can be represented by a matrix A as $y=A\cdot x$, where, x is 32-bit input and y is 32-bit output. Now, assume that the output difference Δy is obtained from Δx as [133],

$$y+\Delta y = A\cdot(x+\Delta x) = A\cdot x + A\cdot\Delta x \Rightarrow \Delta y = A\cdot\Delta x \tag{6-46}$$

On the other hand, the input mask c_x is obtained from the output mask c_y as [133],

$$c_y^T\cdot y = c_y^T\cdot\left(A\cdot x\right) = \left(A^T\cdot c_y\right)^T\cdot x \Rightarrow c_x = A^T\cdot c_y \tag{6-47}$$

From (6-46) and (6-47), the minimum number of linearly *active GSs* of f_1 is equivalent to the minimum number of differentially *active GSs* of f_1 with any (4,4)-BP. Therefore, it's enough to compute the minimum number of differentially *active GSs* of f_1 to find a bound on the minimum number of linearly *active GSs*.

The following simple algorithm could be performed to generate a sample of 1050 different inner functions f_1:

Step	**Algorithm.1: Generating 1050 different f_1**
1	T=0;
2	Choose randomly a set A_{GS} of *GS*s from the all GS-classes
3	Choose randomly a set B_{Odd} of odd different values of b between 1 and 255
4	Select randomly one (4,4)-BP from 6 generated BPs.
5	Generate one f_1 by using only one chosen GS: $GS \xleftarrow{U} A_{GS}$, one value of b: $b \xleftarrow{U} B_{Odd}$, and the selected (4,4)-BP
6	T=T+1;
7	Repeat 5 and 6 till T=1050.

An exhaustive search has been performed to compute the number of differentially *active GSs* for the previous sample of f_1 inner functions in WCS using the following properties:

- The right subblocks: $R=(R_1, R_2, R_3, R_4)$, where, $R \in \mathbb{Z}_{2^{32}}$ and $R_j \in \mathbb{Z}_{2^8}$ for j=1,2,3,4.
- Let $\Delta R_1=(\Delta x\|0^4)$ be the all possible difference in WCS, for all $x \in \mathbb{Z}_{2^4}$, where, 0^n denotes the bit block of n zeros. Then the input difference of a generated f_1 is $\Delta R=(\Delta x\|0^4, 0^8, 0^8, 0^8)$.
- If the input difference of GS_i is non-zero, then the output difference will be non-zero.
- Applying ζ-mapping on any bundles containing zero-differential values, it will produce a zero-differential value.

- Applying ζ-mapping on any bundles containing non-zero-differential values such as $aR_1+bG_1(R_2)$, where, $R_1, R_2 \in \mathbb{Z}_{2^8} \setminus \{0\}$, it will produce either a non-zero-differential value or zero-differential value if $aR_1+bG_1(R_2)$ is a multiple of 2^8.

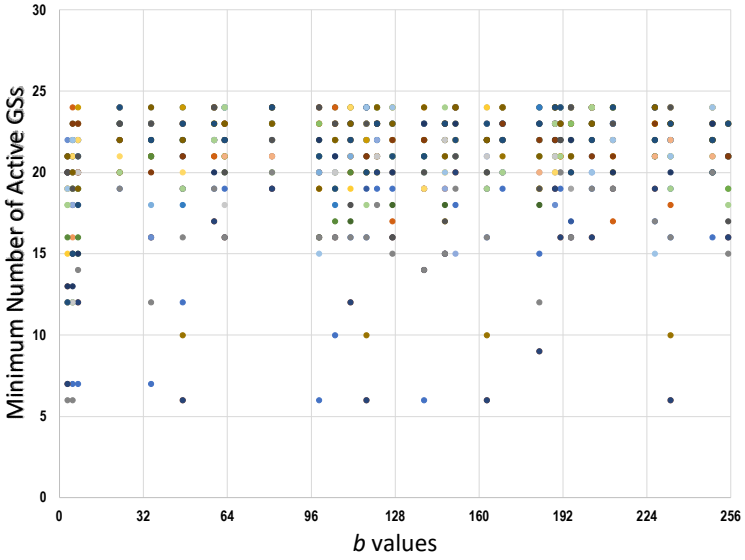


Figure 6-11. Minimum Number of Active GSs with Different Values of *b* in a Differential Trial of 8 Rounds.

In order to present the exhaustive search results, Figure 6-11 shows the minimum number of differentially *active GSs* in 8 rounds of f_1 for different odd values of *b*. Note that the minimum numbers range from 6 and 24 differentially *active GSs.*

Figure 6-12 shows an example of the active GSs though a differential trial in WCS. Here, the output of the subblock R_3 in WCS is zero in the fourth round, due to being a multiple of 2^8. The same case appears in the fifth round where the output of the subblock R_1 in WCS is a multiple of 2^8 as well. To avoid this problem, choosing a different parameter *b* for every ζ-mapping of the inner function f_1 decreases the probability the output of the subblock in any rounds is a multiple of 2^8.

Repeating the same experiment with utilizing several BPs results with the same number of *active GSs*. Table V shows the minimum number of differentially/linearly *active GSs* with r=8 rounds for different values of *b*. From this table, 8 rounds of f_1 have at least 6 *active GSs* at b>1.

Table V. Minimum number of active GSs with r=8 rounds.

b	**3**	**23**	**35**	**47**	**63**	**99**	**147**	**183**	**233**
# active GSs	7	19	7	12	16	6	15	9	16

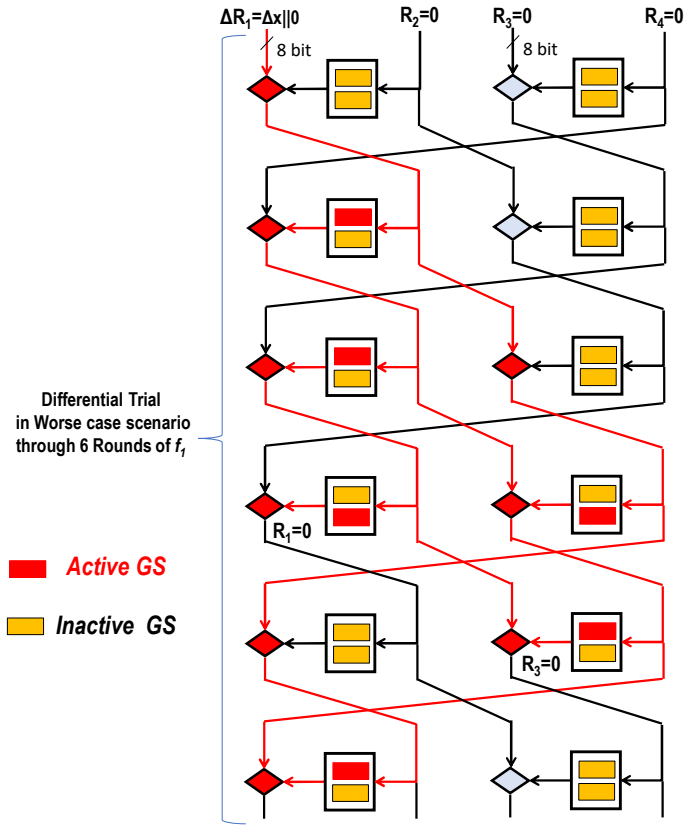


Figure 6-12. The Differential Trial Through f_1 in the Worst-Case Scenario.

6.5. FC2: New Feistel-Like Ciphers-Class Using a Bricklayer Function

The second inner function is more efficient in the targeted implementation platform, where, the second inner function is constructed as a bricklayer function [20]. In particular, the second proposal is simply constructed based on randomly chosen *GS*s connected in parallel as a bricklayer function [20].

6.5.1. Bricklayer Function as a Possible Inner Function Design

One of the simplest architectures of the inner function of the proposed Feistel-like cipher can be considered as a bricklayer function [29]. Here, the proposed bricklayer function can be perceived as a Boolean function that is composed of parallel components or GSs on smaller inputs [20]. However, the proposed bricklayer function is mathematically defined as,

$$f_2(x) = \left(GS_1(x), \ldots, GS_m(x_m)\right) \tag{6-48}$$

Where, $(x_1, \ldots, x_m)$, $n \bmod 4 = 0$, and $x_i \in \{0,1\}^4$: for every $i > 0$.

Figure 6-13 illustrates a Feistel-like cipher with a bricklayer function as an inner function f_2.. Here, FC_2 indicates the second resulting ciphers-class with the second inner function f_2. The class-cardinality $|FC_2|$ is related to the number of cipher rounds t, and the total number of all chosen GSs. That implies,

$$|FC_2| = \left(2^{(n-1)}\right)^t \cdot \mu_2 = 2^{t(n-1)} \cdot 2^{m \times 19.1} = 2^{t \cdot (n-1) + 19.1m} \tag{6-49}$$

Where, μ_2 is the total number of all possible inner function f_2.

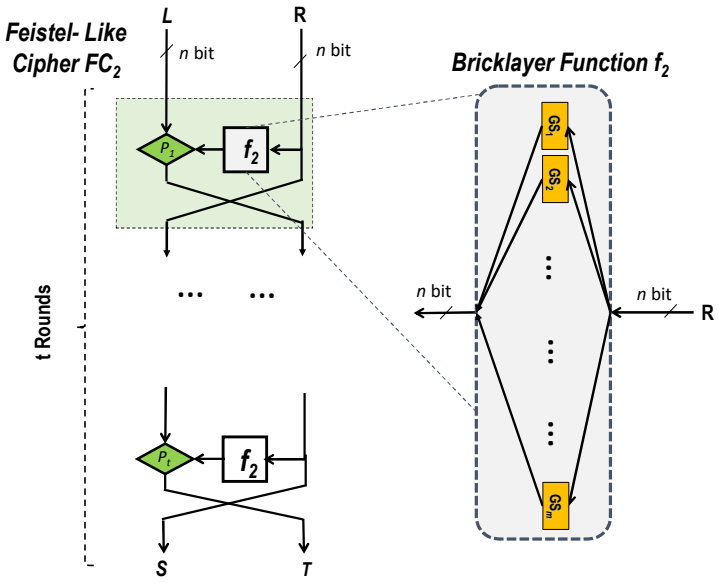


Figure 6-13. Possible Design of the Proposed Cipher Based on the Bricklayer Function Using GSs.

To analyze the security level of the proposed cipher from class FC_2, compute the number of differentially *active GSs* for a sample of f_2 inner functions in WCS, where the input data size is $2n$=64. Note that 8 different GSs are required to construct the inner function f_2 for each cipher as shown in figure 6-14. However, it is assumed that the WCS is defined when only one GS is deployed 8 times to construct the inner function f_2.

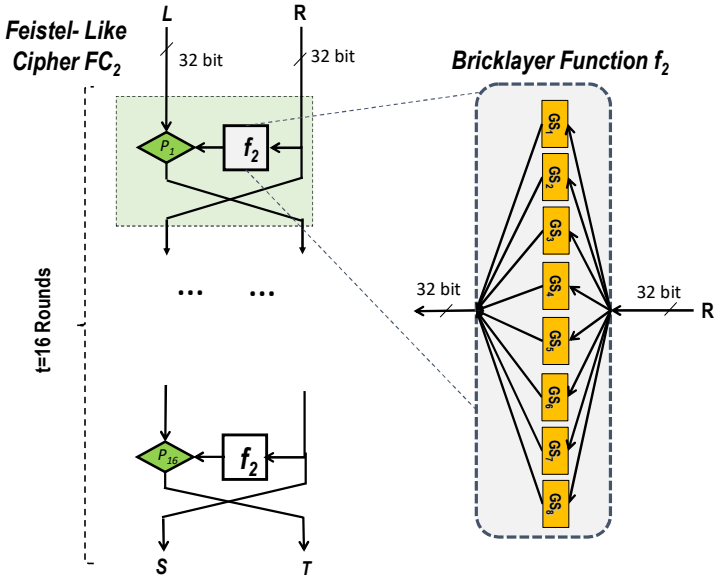


Figure 6-14. A 64-bit SUC as a Feistel-Like Cipher FC_2 [29].

According to (6-49), the class cardinality $|FC_2|$, when each cipher is given as 16-rounds of $\eta(f_2)$ with input size of $2n$=64, is computed as:

$$\left|FC_2\right| = 2^{31\times16}\times2^{8\times19.1} \approx 2^{649} \quad \text{different ciphers} \qquad (6\text{-}50)$$

An exhaustive search was performed to determine the highly likely differential and linear trials for a sample of $10^{4.3}$ different ciphers using the following properties:

- The right subblock is $R = 0$.
- The right subblock is: $L = (L_1, \cdots, L_8)$, where, $L \in \mathbb{Z}_{2^{32}}$ and $L_j \in \mathbb{Z}_{2^4}$ for $j = 1, \ldots, 8$.
- Let $\Delta L_1 = \Delta x$ denote all possible differences, for all $x \in \mathbb{Z}_{2^4}$. Then the input difference of a generated f is $\Delta L = (\Delta x \,||\, 0^{28})$.
- If the input difference of a GS_i is non-zero, then the output difference will be non-zero.
- Applying ζ-mapping on any zero-differential values will produce a zero-differential value.
- Applying ζ-mapping on any non-zero-differential values will produce either a non-zero-differential value or zero-differential value if it is a multiple of 24.

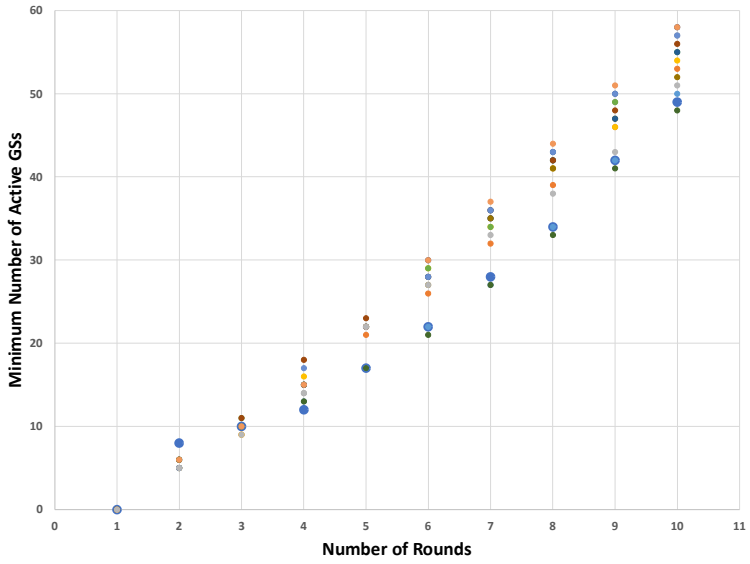


Figure 6-15. Minimum Number of Active GSs in a Differential Trial of 4 Rounds [29].

The results show that 4 rounds of the proposed cipher have at least 12 active *GS*s at $1<b<10000$, where b is an odd number. Figure 6-15 illustrates that the minimum number of differentially active GSs in 4 rounds of the proposed cipher ranges from 12 and 18 differentially active GSs for 20000 different odd values of b in WCS.

Note that after increasing the number of rounds the active GSs increased proportionally. The ciphers having only 12 active GSs after 4 rounds mostly stayed in the bottom in their number of active boxes (marked as bold blue circles) but never divert far away from the remaining sample ciphers. After 10 rounds, at least 48 GSs (out of 80) were active [29].

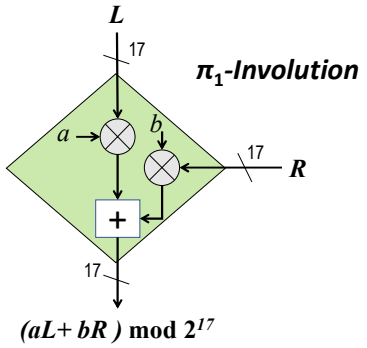
6.6. Hardware Complexity and Possible Implementations

The SUC creation-process is performed by a GENIE that will run in an enrollment process for each unit (See chapter 3). The GENIE should consume small memory, as well as being

simple and fast. Therefore, the memory-performance tradeoff (Space–time tradeoff) should be precisely decided.

6.6.1. π_i-Mappings Hardware Complexity

In the following, πi-Mappings were implemented in Microsemi Smart-Fusion®2. The hardware complexity of each implemented π_i-Mapping was computed based on the number of consumed MACCs and, their used LUT and DFF, where the hardware realization of these mappings is fundamentally performed based on a wide multiplier for size more than 17x17 [79].



Hardware Complexity

n	*LUTs*	*MACCs*
17	17	2
18	18	2

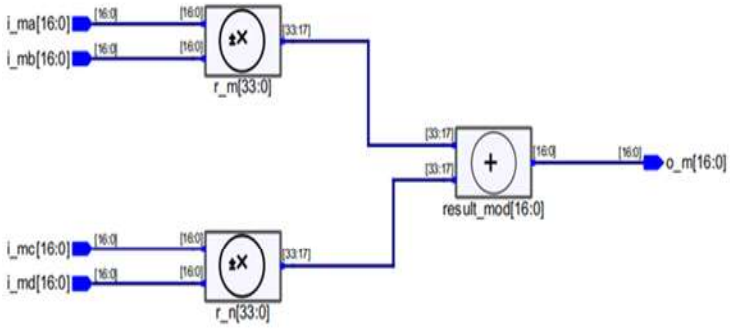
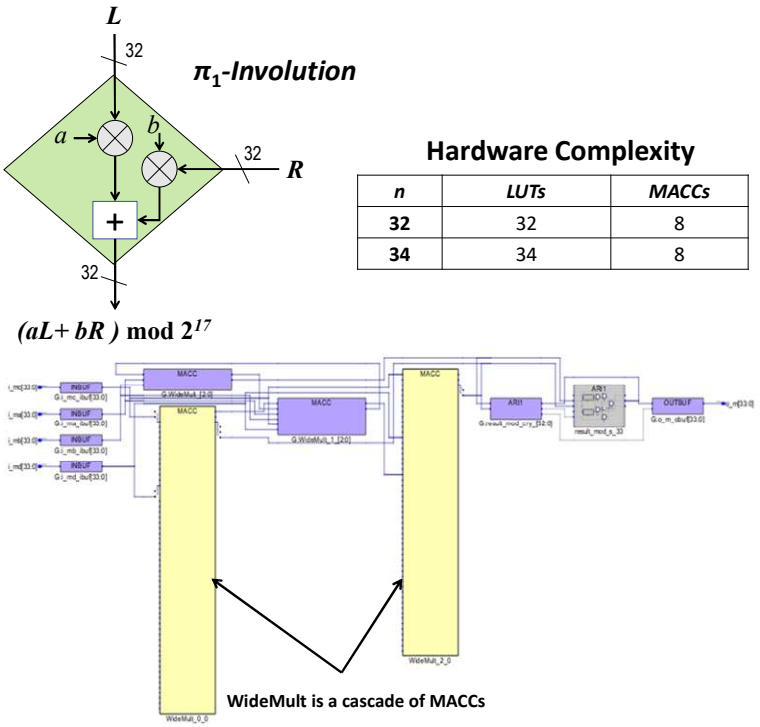


Figure 6-16. FPGA Implementation of π_1-Mapping for 17- and 18-bits Input Size [29].

Figure 6-16 illustrates the resource utilization for π_1-Mapping with n=17 and 18 as bits input size. Note that the consumed resources of π_1-Mapping with n=17 bits input size are 2 MACCs and 17 LUTs. For n= 32 and 34 input size, two wide multipliers were deployed. In this case, each wide multiplier is performed as a cascade of 4 MACCs.

Figure 6-17 shows the required number of MAACs to build two wide multipliers consuming 32 interface-LUTs when n=32 and 34 interface-LUTs when n=34.

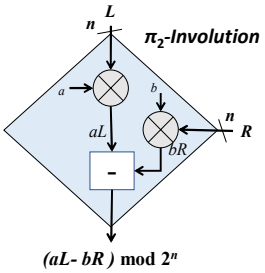


Hardware Complexity

n	LUTs	MACCs
32	32	8
34	34	8

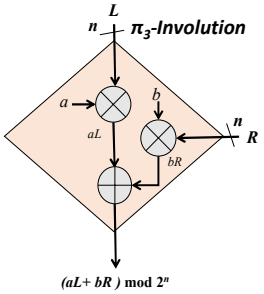
Figure 6-17. FPGA Implementation π_1-Mapping using two Wide Multipliers for 32- and 34-bits [29].

Figure 6-18 shows the required number of MACCs to implement π_2- and π_3-Mappings with the input size of n=17,18,32, and 34. Note that two MACCs are required when n=17 and 18 and two wide multipliers are performed as a cascade of MACCs when n=32 and 34.



π_2- *Mapping* Hardware Complexity

n	LUTs	MACCs
17	18	2
18	19	2
32	33	8
34	35	8



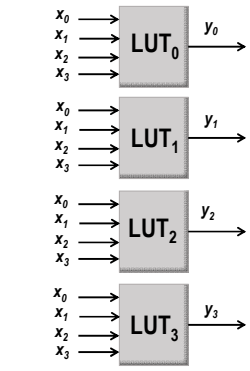
π_3- *Mapping* Hardware Complexity

n	LUTs	MACCs
17	17	2
18	18	2
32	33	8
34	34	8

Figure 6-18. FPGA Implementation of π_2- and π_3-Mappings for n=17-, 18-, 32- and 34-bits Input Size [29].

6.6.2. Ciphers-Class FC_1: A Possible Implementation

In what follows, we present the hardware complexity of the SUCs resulting from the proposed GENIE design strategy.



x_0	x_1	x_2	x_3	y_0	y_1	y_2	y_3
0	0	0	0	0	0	1	0
0	0	0	1	0	1	0	1
0	0	1	0	1	0	1	0
0	0	1	1	1	0	0	0
0	1	0	0	1	0	1	1
0	1	0	1	1	0	0	1
0	1	1	0	0	0	0	1
0	1	1	1	1	1	0	0
1	0	0	0	0	0	1	1
1	0	0	1	0	1	1	0
1	0	1	0	1	1	0	1
1	0	1	1	0	1	0	0
1	1	0	0	1	1	1	1
1	1	0	1	0	0	0	0
1	1	1	1	0	1	1	1

Figure 6-19. The LUT Implementation for *GS* and its Truth Table [99].

The following complexity figures can be considered [99]:

- A 64-bit multiplexer is required for iterative ciphers, where, 64-bit multiplexer requires 64 4-LUTs.
- 64 DFFs are required to store the round state and 32 DFFs are required to store the round of the inner function f_2.
- Each 4-bit GS requires 4 LUTs.

Figure 6-19 illustrates that 4 4-input LUTs are required for implementing a 4×4 GS. The 4×4 mapping $GS = (y_0, y_1, y_2, y_3)$ can be perceived as the parallel application of four binary Boolean functions $y_i = h_i(x_0, x_1, x_2, x_3)$, where $i = 0,1,2,3$. As a result, the implementation of any one GS requires 4 LUTs.

To implement a possible compact version of a selected cipher from FC_1 with input size $2n$=64 bits, the architecture of Figure 6-20 is proposed as a successive round-based implementation [134]. Iterating one cipher-round $\eta(f)$ is utilized as the key idea of the hardware structure. The state machine structure is efficiently completed to run the 16-cipher rounds by deploying a 64-bit register together with a 64-bit multiplexer, where, each cipher-round is executed in one clock-cycle [80]. Here, additional 6 clock-cycles are required every round to compute the 6 rounds of the inner function. Furthermore, a new technique of key scheduling was presented in [80] storing the 16-round-keys in 31 LUTs as shown in Figure 6-20. The round-keys are fully randomly chosen from the TRNG by the GENIE.

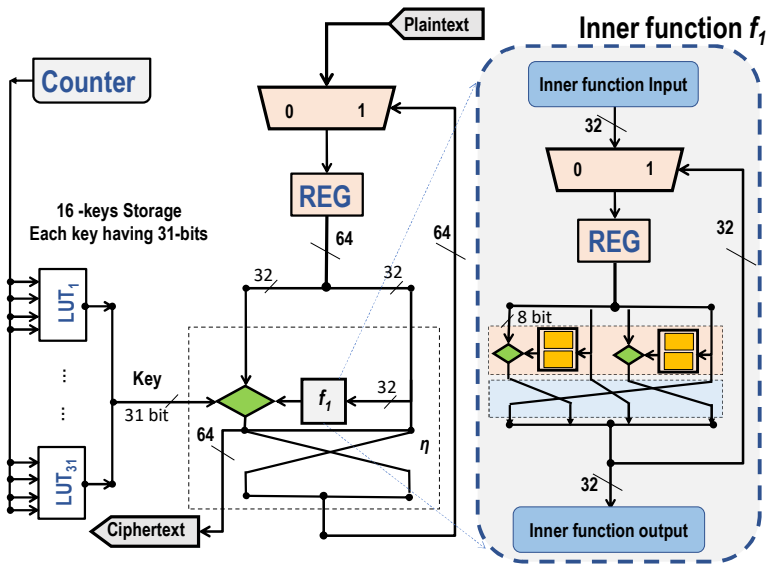


Figure 6-20. A Possible Hardware Architecture of the proposed SUC FC_1 [80].

Table VI shows the hardware implementation complexity of the proposed cipher from FC_1 in a SmartFusion®2 SoC FPGA:

Table VI. FC_1 Hardware Complexity Using SmartFusion®2 M2S025T FPGA.

Hardware Resources	#LUTs	#DFFs	#MACCs
The proposed SUC	176	113	8

6.6.3. Ciphers-Class CF2: A Possible Implementation

The aim of the designed structure is to iterate one cipher-round $\eta(f_2)$, where, a state machine is deployed to run the 16-cipher rounds using a state register of 64-bit and a 64-bit multiplexer, each cipher-round is executed in one clock-cycle [134].

Furthermore, a new technique of key scheduling was presented in [80] storing the 16-round-keys in 31 LUTs as shown in Figure 6-21. The keys are arbitrarily and randomly chosen by the GENIE.

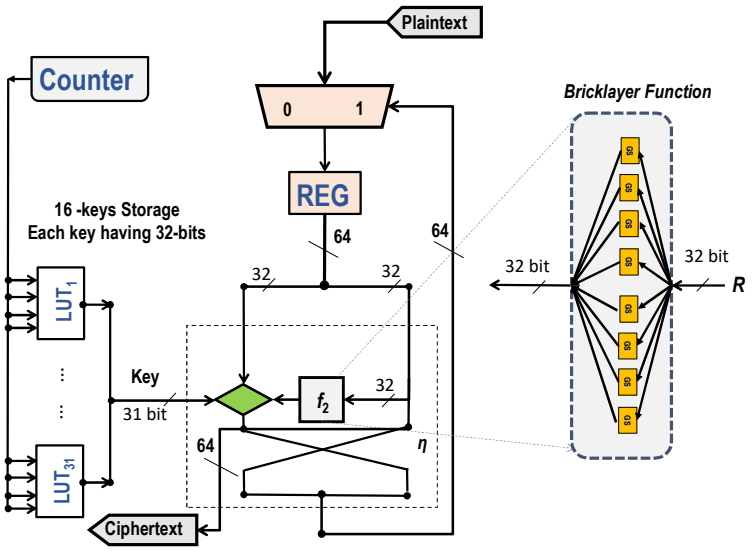


Figure 6-21. A Possible Hardware Architecture of the proposed SUC FC_2 [29].

Table VII illustrates the resulting complexity of the hardware implementation of the proposed cipher from FC_2 in a SmartFusion®2 SoC FPGA. Further, more optimized implementations are under investigations.

Table VII. FC_2 Hardware Complexity Using SmartFusion®2 M2S025T FPGA [29].

Hardware Resources	*LUTs*		*DFFs*		*#MACCs*
	#LUTs	*%*	*#DFFs*	*%*	
SUC using π_1	174	0.62	70	0.25	8
SUC using π_2	175	0.63	70	0.25	8
SUC using π_3	175	0.63	70	0.25	8

Figure 6-22 describes the hardware layout of the cipher's round function. The round's input data X is 64 bits divided into two subblocks of 32-bits. The wide multiplier (Mult_1) is performed to multiply left subblock X[63:32] by a constant a[31:0].

On the other hand, the 8 parallel GSs as a Bricklayer function is applied on the right subblock X[31:0]. Then, the wide multiplier (Mult_2) is performed to multiply the resulting 32-bits by a 32-bits derived from the key storage as a b[31:0]. After that, the outputs of the two wild multipliers are XORed together, then the first 32-bits of the resulting XORed data are concatenated with the right subblock X[31:0] to get the round's output data [63:0]. Here, the round's output data of 64 bits is considered as a new state for the next round. Note that the 16-rounds of the proposed cipher are using the same round function resources each time with a different 32-bit-key.

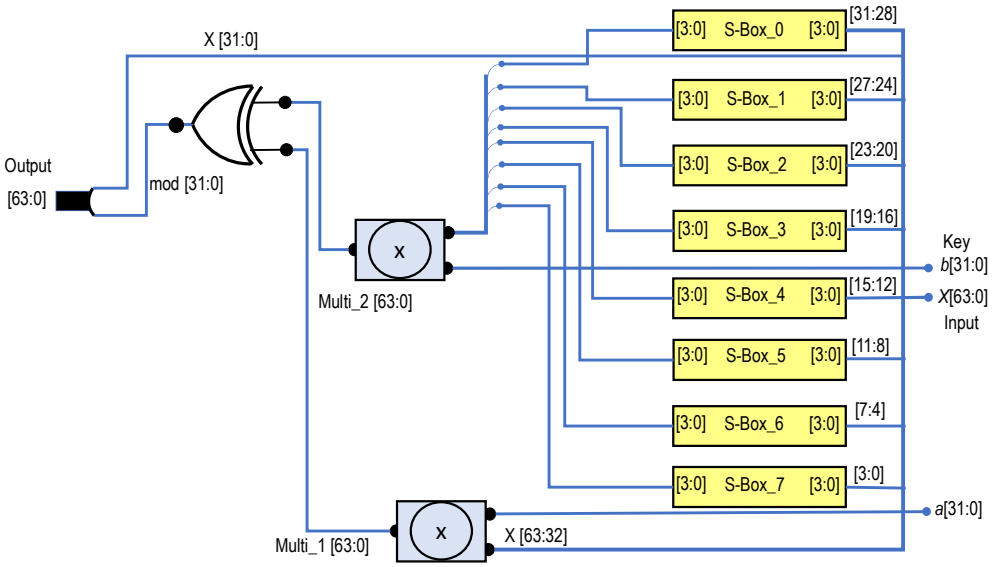


Figure 6-22. Hardware Layout of the Round Function f_2.

The implementation is just aiming to evaluate the hardware complexity. A real implementation procedure is currently not possible as Microsemi is not allowing self-reconfiguration in its current devices. This is expected in future device generations.

6.7. Security Analysis and Evaluation

In this section, modeling attacks on the proposed classes are discussed and the security level of the proposed SUC is evaluated by using the cryptanalysis of a cipher with secret components. Then, quantum exhaustive search for SUC-Model is presented.

Let t be the number of rounds of the proposed Feistel-Like cipher with $2n$ input size. It has theoretically $t(n2^n) + n\text{-}1$ secret key size concerning (6-11). Z-bit is a limit on the amount of computation that an adversary can do to analyze q query. That implies, $q \cdot Z$ bits will be known. Such limitation was presented in [66] as Bekenstein bound for processing and memory density.

In this case, an adversary has a non-negligible advantage to distinguish between the proposed Feistel-Like cipher and a random function, if the adversary can analyze (know) amount of the data at least as great as that of secret key (Shannon's theorem). So that,

$$q \,.Z \geq t(n.2^n) + n\text{-}1 \tag{6-51}$$

The increase of the round numbers t leads to increase q query that a bounded adversary needs. However, the main assumption in the distinguishing attacks is that there is no limit on the amount of computation that an adversary can do. Therefore, the adversary is theoretically unbounded and $q \cdot Z$ bits are known [119].

6.7.1. Modeling Attacks on FC_1 and FC_2

In modeling attacks, the adversary tries to construct a ML algorithm which behaves indistinguishably from the original function (such as PUF) on almost all CRPs. According to section 6.2.2, the proposed SUC is a secure PRF. This implies that the output of SUC is statistically independent of $\left(x_1, SUC\left(x_1\right)\right)$, …, $\left(x_q, SUC\left(x_q\right)\right)$ and uncorrelated with any learner. Therefore, there is no ML algorithm that can build a predictive model for such SUCs.

6.7.2. Cryptanalysis of FC_1 as a Block Cipher with Secret Components

For more practical analysis, if the cipher input size is $2n$=64 bits, then $|FC_1| = 2^{582}$ different SUCs are given according to (6-45). The successful attack on SUC occurs if an adversary can identify an SUC from 2^{582} different SUCs. In this case, the adversary needs to recover the randomly chosen GSs, the randomly chosen BP, and the coefficients b_i of ζ- involutions of the SUC. Therefore, the successful prediction of the adversary is possible with a probability,

$$\frac{1}{\left(2^{19.1}\right)^4 \cdot 6 \cdot 2^7 \cdot \left(2^{31}\right)^{16}} \approx \frac{1}{2^{582}} \tag{6-52}$$

Several differential attack scenarios on a block cipher with secret components were proposed in [135]. Most of the proposed attacks are not applicable to SUC- Model, where, the adversary does not have access to the specific round function. In [136], the proposed attack on a block cipher with secret components analyzes only the plaintext-ciphertext pairs to recover the secret cipher-components one by one. According to this attack scenario and assuming that the adversary tries to attack one SUC as Feistel-like cipher, WCS is when only

GSs are unknown components ignoring the round keys as they may be reachable. It is also assumed that the adversary knows the parameters of the SUC without being able to access the rounds inputs and outputs.

Firstly, the proposed attack starts with gathering T pairs of plaintexts of the form [136]:

$$P_{L,r} = \{(L_i, x \,||\, r_j); x \in \mathbb{F}_2^4\} \tag{6-53}$$

Where, $L_i \in \mathbb{F}_2^{32}$, and $r_j \in \mathbb{F}_2^{28}$; for $0 \leq i, j \leq T$. After that, the adversary finds all pairs $\{x,y\}$ from $P_{L,r}$ such that:

$$(L_i, x \,||\, r_j) \oplus (L_i, y \,||\, r_j) = (0^{32}, x \oplus y \,||\, 0^{28}); \quad \text{for } 0 \leq j \leq T \tag{6-54}$$

Then, the adversary determines the counter set $C(\{x,y\})$ based on the corresponding ciphertext difference all r_j as follows,

$$C(\{x,y\}) = \{L_i, r_j \,|\exists k; SUC(L_i, x \,||\, r_j) \oplus SUC(L_i, y \,||\, r_j) = (0^{4k} \,||\, e \,||\, 0^{60-4k})\} \tag{6-55}$$

Where, $e \in \mathbb{F}_2^4$. In order to recover only one GS, the adversary uses $C(\{x,y\})$ to count how often only one active GS is involved in the ciphertext difference. That is equivalent to,

$$e = GS(x) \oplus GS(y) \tag{6-56}$$

Let D_e be the set of all $\{x,y\}$pairs that hold (6-56). According to [135], if hw(e)=1, then finding 4 sets of form of D_e is enough to determine uniquely the targeted GS, finding 3 sets of form of D_e determines 8 possible S-Boxes as candidates etc.

To estimate this attack, let $\Pr_i[\{x, y\}]$ be the probability that the ciphertext difference has i-th active GSs when the plaintext is chosen from $P_{L,r}$. According to the table V, the minimum number of active GSs is 6 in any differential trials through 16 rounds. That implies,

$$\Pr_1[\{x, y\}] \leq \Pr_6[\{x, y\}] = \left(\left(\frac{1}{4}\right)^6\right)^{16} = \frac{1}{2^{192}} \tag{6-57}$$

Furthermore, the attack complexity β can be lower bounded by the number of pairs that requires to find a set of form of D_e with hw(e)=1,

$$2^{192} \leq \beta \tag{6-58}$$

In conclusion, the proposed SUC is sufficiently secure against any adversary who just analyzes the plaintext-ciphertext pairs. The same analysis was performed and published in [31] for the ciphers-class FC_2. The results showed that the probability that the ciphertext difference has only one active GSs, when the plaintext is chosen from $P_{L,r}$, is 2^{-96}.

6.7.3. The GENIE's Complexity Figures

In the following, some explanations about GENIE' work are given, when the GENIE would realize a cipher from the class FC_1, where, the cipher input size is $2n$=64 as delineated in Figure 6-23.

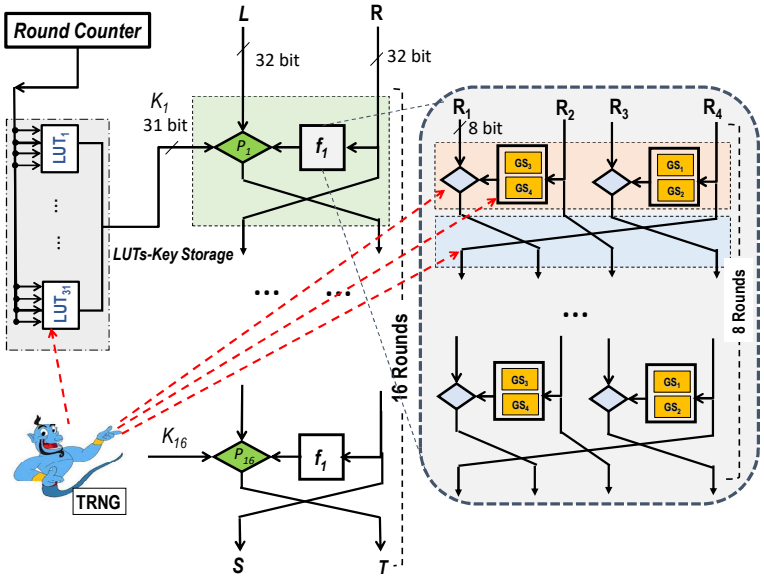


Figure 6-23. A Possible Scenario of GENIE's Work.

Firstly, the GS-generator in (6-35) requires 128 storage bits for each GS-seed, 4 storage bits for each BP, and 16 storage bits for each possible bit-permutation matrix. This implies that the GS generator requires a total of $4\times128+24\times16+6\times4=920$ storage bits for the 4 GS-seeds, 6 BPs, and 24 possible bit-permutation matrices [99].

Secondly, the GENIE randomly generates two functions G_1, G_2 as follows,

- GENIE generates randomly one *GSs* by using (6-35). In this case, a generated GS requires 20 bits, namely, 2 bits for selecting one GS-seed out of 4 GS-seeds, 2x4=8 bits for selecting the parameters $a, b \in \mathbb{F}_2^4$, and 2x5=10 bits for selecting the two permutation matrices P_i, P_j out of all 24 permutation matrices [29].
- GENIE repeats the previous step 4-times to generate randomly 4 *GSs*. Here the GENIE consumes 20×4 GSs = 80 TRNG bits to generate all 4 GSs [29].
- GENIE selects randomly one of the *BPs* from {(4,4)-BP_i }; i=1,2,3,4,5,6. In this case, the GENIE consumes 3 bits for selecting one BP out of 6 BPs.
- GENIE select randomly $r\geq6$ the round number of f_2 (r=8 as a practical example).
- After that, the GENIE consumes 31×2^4=496 TRNG bits for all 16 round keys to be stored in 31 LUTs. A round key is the 31-bits b_i of every $P_i(L,R)=aL+b_iR$; i=1,…,16 [29].
- Eventually, when the GENIE completes FC_2, the GENIE fully and irreversibly deletes itself [29].

In total, the GENIE requires 496+80+3= 579 TRNG bits, and 920 memory storage bits to generate a single cipher choice [99]. Note that a similar approach was published in [29] where the GENIE would realize a cipher from the class FC_2, where, the cipher input size is $2n$=64.

6.7.4. Post Quantum Exhaustive Search Attack

To identify an SUC from all generated FC_2 by using quantum exhaustive search Algorithm such as Grover's algorithm [25], the search space is reduced significantly by a factor of α which is exactly computed as,

$$\alpha = \sqrt{\sigma_2} = \sqrt{2^{t(n-1)} \cdot \mu_2} \approx 2^{291} \tag{6-59}$$

As a consequence, the Feistel-like ciphers can hold up to the proposed quantum attack.

6.8. Summary

In this chapter, a new hardware-oriented cipher design for the proposed concept of SUCs is introduced. The design is deploying hard-core multipliers as major building blocks which are available in modern System on Chip SoC FPGA devices. The ultimate target of the cipher design is to allow converting future SoC FPGA devices into physical clone-resistant units at possibly zero-cost. We assume that zero-cost is attained, if embedding such a SUC module in a device do not consume any area cut from the usual application resources. The resulting complexity of the proposed designs are quite promising consuming less than 1% of the device resources in one of the smallest SoC FPGAs.

Table VIII. Cardinality and Hardware Complexity of the Proposed SUC-Classes

SUC-Class	***SUC-Class Cardinality 2n=64 bits***	***Hardware Resources***				
		#LUTs	***%***	**#DFFs**	***%***	**#MACCs**
***FC*$_1$**	2^{582}	176	0.63	113	0.40	8
***FC*$_2$**	2^{649}	174	0.62	70	0.25	8

In table VIII, the cardinality of the designed ciphers-classes FC_1 and FC_2 are presented and the hardware complexity of the cipher implementation is concluded.

"*Anyone who considers arithmetical methods of producing random digits is, of course, in the state of sin. For, as has been pointed out several times, there is no such thing as a random number—there are only methods to produce random numbers, and a strict arithmetic procedure, of course, is not such a method.*"

"Various Techniques Used in Connection with Random Digits", Journal of Research of the National Bureau of Standards, Appl. Math. Series, Vol. 3 (1951), 3, 36.

John von Neumann (1903-1957)

7. SELF-INVERSE PERMUTATIONS FOR SUCs

In this chapter, new non-algebraic permutation functions are presented. The target usage of such functions is in realizing SUCs. The proposed permutation functions are designed based on the T-Functions technique [126] [137] to create large classes of the round stage functions. The resulting key-entropy of the classes are shown to be cryptographically significant even in a low-complexity processing environment. The hardware and software complexity for realizing such functions are evaluated. Note that it is still possible to deploy the proposed functions for conventional ciphers similar to [138]. The contents of this chapter have been published in [31].

Previously, Shannon introduced two properties, confusion and diffusion, to describe the operation of a secure cipher in [21],. Where, diffusion and confusion are considered as the two essential ciphering requirements that inhibit statistical analysis of the plaintext-ciphertext pairs behavior. Ordinarily, S-Box is designed to serve as a confusion component to the modern cipher [20]. Several proposals of S-Box were proposed such as an Affine-Power-Affine S-Box [139], a gray S-box [140], and a residue prime S-box [141]. On the other hand, a permutation mapping is widely deployed to be a diffusion component such as a fixed wired permutation in PRESENT cipher [142], a bundle permutation of eight 4-bit words in LBlock cipher [103], and a special matrices in AES [20]. Recently, Yunwen *et. al* [138] proposed two types of nonlinear functions as an alternative for a diffusion component. Such functions are not only used to achieve a high level of the diffusion in the designed cipher but also to increase the level of confusion. The first proposed nonlinear function relies on a nonlinear Kerdock code. The second type relies on T-functions.

7.1. Preliminaries on Crypto-Permutations

In [125], Rivest introduced the exact conditions required to find permutation polynomials modulo 2^n. Moreover, Singh *et. al* [143] generalized the conditions for permutation polynomials over $\mathbb{Z}_{p^n}$, where p is a prime integer. The inverse of a quadratic permutation polynomial over $\mathbb{Z}_{2^n}$ is investigated in [144] [145]. In [146], the essential conditions for creating self-inverse quadratic permutation polynomials over $\mathbb{Z}_{2^n}$ were determined. On the other hand, an interesting work by Klimov [126] [137] introduced the bit-slice analysis method to create the so-called T-functions. Using this method, Klimov *et .al* [126] proposed more generalized polynomial structures with integer coefficients using (+), (-) and the Boolean operators over 2^n. They obtained the necessary and sufficient conditions for such polynomial structures to become permutation polynomial functions. Following this work, several classes of self-inverse quadratic permutation polynomial functions are designed based on T-Functions technique. The desired permutation functions can be gradually defined as follows:

Definition 7.1: Let R be a finite ring, where, not every function $f : R \to R$ is representable by a polynomial over the same ring R. A function f over R is called a polynomial function if there is a polynomial $P(x) \in R[x]$ such that,

$$f(x) = P(x) \qquad \forall x \in R \tag{7-1}$$

If a polynomial function f is a one-to-one mapping, then it is called a permutation polynomial (PP) over R.

In [125], Rivest presented the exact characterization of such PPs modulo 2^n: n >1.

Theorem 7.1: (Rivest [125]): "A polynomial $P(x)=a_0+a_1x+...+a_d\ x^d$, with integral coefficients is a permutation polynomial modulo 2^n: n >1, if, and only if, a_1 is odd, ($a_2+a_4+\ldots$) is even, and ($a_3+a_5+\ldots$) is even".

For instance, a polynomial $P(x)=ax+bx^2$ is a Quadratic Permutation Polynomial (QPP) over $\mathbb{Z}_{2^n}$, if a is odd and b is even. The following definition clarifies the concept of the self-inverse function.

Definition 7.2: A self-inverse function or an involution is a function f that satisfies:

$$f(f(x)) = x \tag{7-2}$$

for all x.

The following lemma gives the conditions for QPP to be a self-inverse quadratic permutation polynomial (SIQPP).

Lemma 7.2: [146] Let n>2 be an integer and let $P(x)=ax+bx^2$ be a QPP over $\mathbb{Z}_{2^n}$. Then, $P(x)=ax+bx^2$ is self-inverse if, and only if, the following conditions hold true:

1. $a = -1 + 2^{n-1} \cdot u$, where u is a unit in $\mathbb{Z}_{2^n}$.
2. If n is even, then $b = 2^r \cdot v$, where: $r \geq n/2$ and v is an unit in $\mathbb{Z}_{2^n}$.
3. If n is odd, then $b = 2^r \cdot v$, where: $r \geq (n-1)/2$ and v is an unit in $\mathbb{Z}_{2^n}$.

Unfortunately, each permutation polynomial from the previous class has not the required strong cryptographic properties. In fact, there is a weakness in that at least two fixed points exist in the resulting permutation for every SIQPP over $\mathbb{Z}_{2^n}$. The following lemma proves that every SIQPP from the previous class has at most two fixed points.

Lemma 7.3: Any $P(x)=ax+bx^2$ satisfying the conditions of theorem 7.2, has exactly two fixed points.

Proof:

The first fixed point, as with all self-inverse permutation polynomial of the previous class, is x=0. To determine the second fixed point, assume that n is an even number (odd cases can be proved similarly) and $x=2^{n-1}$,

$$P(2^{n-1}) = a(2^{n-1}) + b(2^{n-1})^2 \tag{7-3}$$

So,

$$P(2^{n-1}) = (-1 + 2^{n-1}u)(2^{n-1}) + (2^r v)(2^{n-1})^2 \tag{7-4}$$

Where, $r \geq n/2$, and u and v are units in $\mathbb{Z}_{2^n}$. Thus,

$$P(2^{n-1}) = -2^{n-1} + 2^{2n-2}u + 2^{2n+r-2}v \tag{7-5}$$

Applying mod 2^n on both sides results with,

$$P(2^{n-1}) = 2^n - 2^{n-1} = 2^{n-1}(2-1) = 2^{n-1} \tag{7-6}$$

Which proves that $x=2^{n-1}$ is the second fixed point.

To prove that every self-inverse permutation polynomial of the previous class has no more than the two fixed points $x=0,2^{n-1}$, suppose that x_0 is another fixed point such that $x_0 \neq 0,2^{n-1}$. Then,

$$P(x_0) = a(x_0) + b(x_0)^2 = x_0 \tag{7-7}$$

After substitution,

$$(-1 + 2^{n-1}u)x_0 + (2^r v){x_0}^2 = x_0 \tag{7-8}$$

Thus,

$$2^{n-1}u \cdot x_0 + 2^r v \cdot x_0^2 = 2x_0 \tag{7-9}$$

Multiplying both sides by 2 results with,

$$2^n u \cdot x_0 + 2^{r+1} v \cdot x_0^2 = 2^2 x_0 \tag{7-10}$$

or,

$$2^n u \cdot x_0 + 2(2^r v \cdot x_0 - 2)x_0 = 0 \tag{7-11}$$

By applying mod 2^n on both sides results with,

$$(2^r v \cdot x_0 - 2)x_0 = 0 \tag{7-12}$$

This is only possible if x_0=0 or $x_0=(2^{r-1}v^{-1})^{-1}$ mod 2^n, however $2^{r-1}v$ is not an invertible element over $\mathbb{Z}_{2^n}$. Therefore, x_0=0 contradicts with the assumption $x_0 \neq 0,2^{n-1}$. As a result, every self-inverse permutation polynomial of the previous class has no more than just the above presented two fixed points $x=0,2^{n-1}$. □

7.1.1. T-Functions

In 2003, Klimov and Shamir [126] introduced a new class of low-complexity functions which are invertible and exhibiting special properties and called them Triangular-Functions (T-function).

For $x \in \mathbb{Z}_{2^n}$, a binary representation of x as n-bit vectors is given as follows:

$$x=[x]_{n-1}\ldots[x]_1[x]_0 \quad \text{where, } [x]_i \in \mathbb{Z}_2 \text{ for every } 0 \le i \le n.$$

Definition 7.3 [126]**:** A function $f(x)$ from a n-bit input to a n-bit output with the property that the i^{th} bit of its output depends only on the first, the second... and the i^{th} bit of its inputs is called a T-function (short for triangular function).

Eight basic possible constructing operations of T-functions were introduced as[126]:

- Negation $(-a)$ mod 2^n, Addition $(a+b)$ mod 2^n.
- Subtraction $(a-b)$ mod 2^n, Multiplication $(a.b)$ mod 2^n.
- The Boolean functions: Complement $\overline{a}$, OR $(a \vee b)$, AND $(a \wedge b)$, and XOR $(a \oplus b)$.

Where, a and b are two n-bit words.

The following lemma gives an abstract bit-slicing representation of the arithmetic and logic operations,

Lemma 7.4: For i>0, and $x, y \in \mathbb{Z}_{2^n}$,

$$\left.\begin{array}{l} [x \times y]_0 = [x]_0 \wedge [y]_0,\ [x \times y]_i = [x]_i \alpha_{[y]_0} \oplus [y]_i \alpha_{[x]_0} \oplus \alpha_{x \times y} \\ [x \pm y]_0 = [x]_0 \oplus [y]_0,\ [x \pm y]_i = [x]_i \oplus [y]_i \oplus \alpha_{x \pm y} \\ [x \oplus y]_0 = [x]_0 \oplus [y]_0,\ [x \oplus y]_i = [x]_i \oplus [y]_i \\ [x \overset{\wedge}{\vee} y]_0 = [x]_0 \overset{\wedge}{\vee} [y]_0,\ \left[x \overset{\wedge}{\vee} y\right]_i = [x]_i \overset{\wedge}{\vee} [y]_i \end{array}\right\} \tag{7-13}$$

Where, α's denotes a parameter.

In [126], Klimov generalized Rivest's construction of PPs resulting with invertible mappings with T-functions properties as follows:

Theorem 7.5 [126]*:* "Let $P(x) = a_0 \overset{\oplus}{+} a_1 x \overset{\oplus}{+} \cdots \overset{\oplus}{+} a_d x^d$ be a generalized polynomial with integral coefficients. Then $P(x)$ defines a permutation polynomial modulo 2^n: n >2, if, and only if, a_1 is odd, (a_2+a_4+…) is even, and (a_3+a_5+…) is even".

Definition 7.4: "A T-function $f(x)$ is invertible if it can be represented in the following form:

$$[f(x)]_i=[x]_i +\alpha \tag{7-14}$$

Which holds true for any bit *i, where,* α is a parameter"[147].

The main target of the subsequent sections is to construct T-functions with SIQPF properties deploying the operations (+), (-), and $\oplus$. Therefore, it is necessary to determine the required cryptographically relevant conditions on $P(x) = ax \overset{\oplus}{+} bx^2$ or $P(x) = a \overset{\oplus}{+} bx \overset{\oplus}{+} cx^2$, such that $P(x)$ becomes a SIQPF over the ring $\mathbb{Z}_{2^n}$ and $P(x)$ exhibit different fixed points for different permutations.

7.2. New Classes of Self-Inverse Permutation Functions

In the following, it is shown how to construct QPPs deploying +, -, and $\oplus$ operations to come up with SIQPFs.

Definition 7.5: The functions

$$P(x) = ax \overset{\oplus}{+} bx^2 \tag{7-15}$$

or

$$P(x) = a \overset{\oplus}{+} bx \overset{\oplus}{+} cx^2 \tag{7-16}$$

are said to be Quadratic Permutation Functions (QPFs) over $\mathbb{Z}_{2^n}$, if they permute all the elements of $\mathbb{Z}_{2^n}$.

Definition 7.6: A self-inverse function is a function f over $\mathbb{Z}_{2^n}$, such that,

$$f(f(x)) \bmod 2^n=x \tag{7-17}$$

where x is a n-bit word.

Choosing the coefficients a, b according to lemma 6.2, a, b can be represented as n-bit vectors as follows,

$$a = [a]_{n-1} \cdots [a]_1[a]_0 = 0\underbrace{1\cdots1}_{n-1}, \text{ and } b = \begin{matrix}1\\0\end{matrix} \cdots \begin{matrix}1\\0\end{matrix}\underbrace{0\cdots0}_{r} \tag{7-18}$$

where $r \geq n/2$ if n is even and $r \geq (n-1)/2$ if n is odd.

Lemma 7.6: For n >2, a polynomial $P(x) = a \overset{\oplus}{+} bx \overset{\oplus}{+} cx^2$ is a permutation polynomial over $\mathbb{Z}_{2^n}$ if, and only if, b is odd and c is even (according to theorem 7.5).

Lemma 7.7: In $\mathbb{Z}_{2^n}$;

- If $a = 0\underbrace{1\cdots1}_{n-1}$, then $a^2 \bmod 2^n = \underbrace{0\cdots0}_{n-1}1 = 1$, and $(a+1) \bmod 2^n = 1\underbrace{0\cdots0}_{n-1} = 2^{n-1}$.
- If $a = \underbrace{1\,1\cdots1}_{n}$, then $a^2 \bmod 2^n = \underbrace{0\cdots0}_{n-1}1 = 1$, ($a$+1) mod$2^n$=0, and ($a.u$)mod$2^n$=$2^n$-$u$.
- If $u = \begin{matrix}1\\0\end{matrix} \cdots \begin{matrix}1\\0\end{matrix} \underbrace{0\cdots0}_{i}$ and $v = \begin{matrix}1\\0\end{matrix} \cdots \begin{matrix}1\\0\end{matrix} \underbrace{0\cdots0}_{j}$ are two even numbers, then $u \cdot v = \begin{matrix}1\\0\end{matrix} \cdots \begin{matrix}1\\0\end{matrix} \underbrace{0\cdots0}_{i+j}$, where $i, j \le n$.
- If $b = \begin{matrix}1\\0\end{matrix} \cdots \begin{matrix}1\\0\end{matrix} \underbrace{0\cdots0}_{r}$ is an even number, then b^k mod 2^n=0 for k. $r \ge n$.

The proof is very simple. It is based on the definitions of the multiplication and the modulus over $\mathbb{Z}_{2^n}$.

Now, the following theorem can be proved based on the previous lemma,

Theorem 7.8: Let n >2 be an integer. The permutation function $P(x) = ax \overset{\oplus}{+} bx^2$ defined over $\mathbb{Z}_{2^n}$ is self-inverse if the following conditions are satisfied:

1. $a = 0\underbrace{1\cdots1}_{n-1}$ or $a = \underbrace{1\,1\cdots1}_{n}$ in $\mathbb{Z}_{2^n}$.
2. For n even, $b = \begin{matrix}1\\0\end{matrix} \cdots \begin{matrix}1\\0\end{matrix} \underbrace{0\cdots0}_{r}$ in $\mathbb{Z}_{2^n}$ where $r \ge n/2$.
3. For n odd, $b = \begin{matrix}1\\0\end{matrix} \cdots \begin{matrix}1\\0\end{matrix} \underbrace{0\cdots0}_{r}$ in $\mathbb{Z}_{2^n}$ where $r \ge (n-1)/2$.

Proof:

First, it is required to prove that: If n is even, then $P(x) = ax \overset{\oplus}{+} bx^2$ is a SIQPF over $\mathbb{Z}_{2^n}$, where $a = 0\underbrace{1\cdots1}_{n-1}$. (Other cases can be proved in a similar way).

Let,

$$P(P(x))=P(ax+bx^2) \tag{7-19}$$

and,

$$P(P(x))=a(ax+bx^2)+b(ax+bx^2)^2 \tag{7-20}$$

so,

$$P(P(x))=a^2x+ ab\ (1+a)x^2+2ab^2x^3+b^3x^4 \quad (7\text{-}21)$$

From lemma 7.7 and by applying mod 2^n on both sides results with:

The first term is $(a^2x) \bmod 2^n=1.x=x$,

the second term is $(ab(1+ab)\ x^2) = (ab2^{n-1}x) \bmod 2^n=0$, where$(n\text{-}1) +n/2> n$,

the third term is $(2ab^2x^3) \bmod 2^n=0$, where $2r \geq 2\cdot(n/2) = n$,

and the fourth term is $(b^3x^4) \bmod 2^n=0$, where $3r> 3.n/2>n$.

This implies, that

$$P(P(x)) \bmod 2^n=x \quad \text{for any } n. \qquad \square.$$

The same theorem for the formula in (7-16) can be correspondingly proved and stated as follows:

Theorem 7.9: Let n >2 be an integer. $P(x) = a \overset{\oplus}{+} bx \overset{\oplus}{+} cx^2$ is a self-inverse permutation function defined over $\mathbb{Z}_{2^n}$ if the following conditions are satisfied:

1. $b = 0\underbrace{1\cdots1}_{n-1}$ or $b = \underbrace{1 1\cdots1}_{n}$ in $\mathbb{Z}_{2^n}$.
2. For n even, $a = \begin{matrix}1\\0\end{matrix}\cdots\begin{matrix}1\\0\end{matrix}\underbrace{0\cdots0}_{i}$ and $c = \begin{matrix}1\\0\end{matrix}\cdots\begin{matrix}1\\0\end{matrix}\underbrace{0\cdots0}_{j}$ in $\mathbb{Z}_{2^n}$ where $i, j \geq \frac{n}{2}$.
3. For n odd, $a = \begin{matrix}1\\0\end{matrix}\cdots\begin{matrix}1\\0\end{matrix}\underbrace{0\cdots0}_{i}$ and $c = \begin{matrix}1\\0\end{matrix}\cdots\begin{matrix}1\\0\end{matrix}\underbrace{0\cdots0}_{j}$ in $\mathbb{Z}_{2^n}$ where $i, j \geq \frac{n-1}{2}$.

Proof:

Firstly, it is required to prove that: If n is even, then $P(x) = a \overset{\oplus}{+} bx \overset{\oplus}{+} cx^2$ is a self-inverse permutation function over $\mathbb{Z}_{2^n}$, where, $b = 0\underbrace{1\cdots1}_{n-1}$. (Other cases can be proved in a similar fashion).

Let $g(x)$ be a function which is defined as,

$$g(x)=P(x)-a=bx+cx^2 \quad (7\text{-}22)$$

From theorem 7.8, $g(x)$ is a SIQPF.

To prove the self-inverse property of $P(x)$, further suppose that,

$$P(P(x))=P(a+bx+cx^2) \quad (7\text{-}23)$$

and,

$$P(P(x))=a+ b(a+bx+cx^2)+c(a+bx+cx^2)^2 \quad (7\text{-}24)$$

Yielding,

$$P(P(x)) = a+ ab+ b\ (bx+cx^2)+a^2c+2ac\ (bx+cx^2)+c(a+bx+cx^2)^2 \quad (7\text{-}25)$$

by further simplification,

$$P(P(x)) = a(1+b+ac)+2ac\ (bx+cx^2)+g(g(x)) \qquad (7\text{-}26)$$

Applying mod 2^n on both and from lemma 7.7 sides results with:

The first term: $a(1+b+ac) \bmod 2^n=(a2^{n-1}+a^2c) \bmod 2^n$,

where, $i+(n-1)>n/2+(n-1)>n$, and $2i+j>n+n/2>n$,

the second term: $2ab(bx+cx^2) \bmod 2^n=0$,

where, $ac=0$ for $i+j>n/2+n/2=n$,

and the third term: $g(g(x)) \bmod 2^n=x$.

This implies, that,

$$P(P(x)) \bmod 2^n = x \quad \text{for any integer } n \qquad \square.$$

Unfortunately, the $P(x)$ class of theorem 7.8 suffers from the same fixed-points weakness as $P(x)$ of theorem 7.2. In other words, two specific fixed points ($x=0,2^{n-1}$) for every SIQPF exist. To overcome such weakness, it is proposed that in this work to remove the specific constant fixed points in theorem 7.8. This is attained similarly as in [137] by deploying the Boolean operator (OR) as proved in the following theorem.

Theorem 7.10: For $n>2$ a polynomial $P(x) = ax \overset{\oplus}{+} b(x^2 \vee D)$ is a SIQPF in $\mathbb{Z}_{2^n}$ if it satisfies all the following conditions:

1. $a = 0\underbrace{1\cdots1}_{n-1}$ or $a = \underbrace{11\cdots1}_{n}$ in $\mathbb{Z}_{2^n}$.
2. For n even, $b = \begin{matrix}1 & & 1\\ 0 & \cdots & 0\end{matrix}\underbrace{0\cdots0}_{r}$ in $\mathbb{Z}_{2^n}$, where, $r \geq n/2$.
3. For n odd, $b = \begin{matrix}1 & & 1\\ 0 & \cdots & 0\end{matrix}\underbrace{0\cdots0}_{r}$ in $\mathbb{Z}_{2^n}$ where $r \geq (n-1)/2$.

D is an integer number.

Moreover, the resulting two fixed points of any $P(x)$ are distinct and different for each individual SIQPF.

Proof:

In the following it will be proved that $P(x) = ax \overset{\oplus}{+} b(x^2 \vee D)$ is a SIQPF, where, n is an even number and $a = 0\underbrace{1\cdots1}_{n-1}$ in $\mathbb{Z}_{2^n}$.(Other cases can be proved in a similar way). According to theorem 7.8 and 7.9, the function $f(x)=ax+bg(x)$ is a SIQPF, where, $g(x)=x^2$. This is always true for $g(x) = x^2 \vee D$, if $f(x)=ax+bg(x)$ is still an invertible function. To check if $f(x)=ax+bg(x)$ is an invertible function the bit-slice method (lemma 7.4) can be used as follows:

$$[f(x)]_0 = [ax + b \cdot g(x)]_0 = [ax]_0 \oplus [b \cdot g(x)]_0 \quad (7\text{-}27)$$

And,

$$[f(x)]_0 = ([a]_0 \wedge [x]_0) \oplus ([b]_0 \wedge [g(x)]_0) = [x]_0 \quad (7\text{-}28)$$

where, $[a]_0$=1, and $[b]_0$=1.

Now, for i >0,

$$[f(x)]_i = [ax + b \cdot g(x)]_i = [ax]_i \oplus [b \cdot g(x)]_i \quad (7\text{-}29)$$

and,

$$[f(x)]_i = ([a]_i \alpha_{[x]_0} \oplus [x]_i \alpha_{[a]_0} \oplus \alpha_{a \cdot x}) \oplus ([b]_i \alpha_{[g(x)]_0} \oplus [g(x)]_i \alpha_{[b]_0} \oplus \alpha_{b \cdot g(x)}) \quad (7\text{-}30)$$

Firstly, the case of 0< i < n/2 is checked:

$$[f(x)]_i = (\alpha_{[x]_0} \oplus [x]_i \alpha_{[a]_0} \oplus \alpha_{a \cdot x}) \oplus (\oplus \alpha_{b \cdot g(x)}) \quad (7\text{-}31)$$

and,

$$[f(x)]_i = [x]_i \oplus \beta \quad (7\text{-}32)$$

where, $\beta = \alpha_{[x]_0} \oplus \alpha_{a \cdot x} \oplus \alpha_{b \cdot g(x)}$.

Now, for $i \geq n/2$,

$$[f(x)]_i = (\alpha_{[x]_0} \oplus [x]_i \alpha_{[a]_0} \oplus \alpha_{a \cdot x}) \oplus ([b]_i \alpha_{[g(x)]_0} \oplus \alpha_{b \cdot g(x)}) \quad (7\text{-}33)$$

and,

$$[f(x)]_i = [x]_i \oplus \gamma \quad (7\text{-}34)$$

where, $\gamma = \alpha_{[x]_0} \oplus \alpha_{a \cdot x} \oplus [b]_i \alpha_{[g(x)]_0} \oplus \alpha_{b \cdot g(x)}$.

From (7-28), (7-32), (7-34) and definition.7.4, the function $f(x) = ax + bg(x)$ can be represented as,

$$[f(x)]_i = [x]_i \oplus \alpha \quad (7\text{-}35)$$

This holds true for any bit i and α is a parameter. Therefore, $P(x) = ax \overset{\oplus}{+} b(x^2 \vee D)$ is an invertible function. It's very simple to show that $b((2^{n-1})^2 \vee D)=bD$ and $b((0)^2 \vee D)=bD$ over $\mathbb{Z}_{2^n}$, therefore, x=0,2^{n-1} are not fixed points for any $D \neq 0$.

Suppose now that for $l = 1, 2$, there are two different SIQPFs, such as $P_l(x) = ax + b(x^2 \vee D_l)$, where $D_1 \neq D_2 \neq 0$, having the same fixed point x_0 as follows:

$$P_1(x_0) = P_2(x_0) \quad (7\text{-}36)$$

which implies, that:

$$x_0^2 \vee D_1 = x_0^2 \vee D_2 \tag{7-37}$$

The last step is only correct, if, and only if, $D_1 = D_2$. This proves that the resulting fixed points of any $P(x)$ are distinct and different for each individual SIQPF. □.

The fact that the resulting SIQPF, according to the construction in Theorem 7.10, gives rise to two individual and different fixed points for each different SIQPF, is advantageous for cryptographic applications. The reason is that, ciphering operations usually involve cascading many different SIQPFs as round functions (see cipher structure in section 7.4 and Figure 7-2). Therefore, the dynamic distribution of the different fixed points for different SIQPFs in different cascading stages generally results in an improved random diffusion property of the overall cipher permutation.

Extending the $P(x)$ of class $P(x) = a \overset{\oplus}{+} bx \overset{\oplus}{+} c(x^2 \vee D)$ in (7-15) similarly as done for the class of function (7-16) in theorem 7.10, results with a new larger class with similar properties as described in the following theorem 7.11:

Theorem 7.11: Let n >2 be an integer, then $P(x) = a \overset{\oplus}{+} bx \overset{\oplus}{+} c(x^2 \vee D)$ is a self-inverse permutation function defined over $\mathbb{Z}_{2^n}$ if the following conditions are satisfied:

1. $b = 0\underbrace{1\cdots1}_{n-1}$ or $b = \underbrace{1 1\cdots1}_{n}$ in $\mathbb{Z}_{2^n}$.

2. For n even, $a = \begin{matrix}1 & & 1\\ 0 & \cdots & 0\end{matrix}\underbrace{0\cdots0}_{i}$ and $c = \begin{matrix}1 & & 1\\ 0 & \cdots & 0\end{matrix}\underbrace{0\cdots0}_{j}$ in $\mathbb{Z}_{2^n}$, where, $i, j \geq n/2$.

For n odd, $a = \begin{matrix}1 & & 1\\ 0 & \cdots & 0\end{matrix}\underbrace{0\cdots0}_{i}$ and $c = \begin{matrix}1 & & 1\\ 0 & \cdots & 0\end{matrix}\underbrace{0\cdots0}_{j}$ in $\mathbb{Z}_{2^n}$, where, $i, j \geq (n-1)/2$.

D is an integer number.

Moreover, the resulting two fixed points of any $P(x)$ are distinct and different for each individual SIQPF.

Proof: The proof can be carried out as described for theorem 7.10.

7.2.1. Cardinality of the Proposed SIQPF Classes

In this section, the cardinality of the SIQPF classes are evaluated. If the designed classes have large cardinalities, then the probability of successful cloning attack or modeling attack approaches zero. Moreover, the equivalent and distinct mappings of the SIQPF s are identified.

In $\mathbb{Z}_{2^n}$, not all permutations can be generated by polynomials and every permutation may be generated by different polynomials which are called equivalent polynomials modulo 2^n. Therefore, the cardinality of a set of distinct polynomial permutations over $\mathbb{Z}_{2^n}$ requires excluding equivalent cases.

Let P_n be a set of all possible permutation polynomials resulting with distinct

permutations over $\mathbb{Z}_{2^n}$. Keller *et al.* [148] presented a formula to determine the cardinality of P_n. The cardinality of the set of all polynomial functions over different rings was presented in [149]. In [150], Jiang obtained a counting function for the number of polynomial functions over general finite commutative ring. Accordingly, the formula of the cardinality of P_n was determined in [148][150] as follows:

$$\left|\mathrm{P}_n\right| = 2^{3+\sum_{k=3}^{n}\beta(k)} \tag{7-38}$$

Where, $\beta(k)$ is the smallest integer s such that 2^k divides $s!$.

Making use of Rivest theorem 7.1, the number of permutation polynomials of degree at most d can be computed by the following lemma:

Lemma 7.12: For n >2, the number of all possible permutation polynomials of degree at most d over $\mathbb{Z}_{2^n}$ is,

$$N_0 = 2^{n+d\cdot(n-1)} \tag{7-39}$$

Proof:

For every n, and from theorem 6.1, the following is always true:

$$a_0 = \underbrace{1\cdots 1}_{n}\underbrace{0\cdots 0}_{} \quad \Rightarrow \left|a_0\right| = 2^n$$

and,

$$a_1 = \underbrace{1\cdots 1}_{n-1}\underbrace{0\cdots 0}_{}1 \quad \Rightarrow \left|a_1\right| = 2^{n-1}$$

By using the fact, that the sum of two even or odd integers is always even and according to theorem 7.1, implies:

$$a_i = \underbrace{1\cdots 1}_{n-1}\underbrace{0\cdots 0}_{}* \quad \Rightarrow \left|a_i\right| = 2^{n-1}$$

Where the * position is a fixed 0 or 1. That is, the number of permutation polynomials which have the form $P(x)=a_0+a_1x+\ldots+a_d x^d$ is:

$$N_0 = 2^n \cdot \underbrace{2^{n-1}\cdots 2^{n-1}}_{d} = 2^{n+d(n-1)}$$ □

Definition 7.7: [151] The polynomials $f(x)$ and $g(x)$ over $\mathbb{Z}_{2^n}$ are equivalent polynomials modulo 2^n, if such polynomials satisfy the following condition:

$$f(x) \equiv g(x) \bmod 2^n \tag{7-40}$$

In other words, the resulting polynomials generates the same permutation.

Note that according to the above lemma, the number of permutation polynomials N_0 may include some equivalent permutation polynomials (EPPs) modulo 2^n. The following definition appears to be useful for the targeted evaluation.

Definition 7.8: The cardinality of the set of all EPPs modulo 2^n with degree $d \leq 2^n$-1, is equal to the numbers of all possible permutation polynomials N_0 having the degree $d \leq 2^n$-1 excluding all distinct permutation polynomials $|P_n|$, thus,

$$|EPPs| = N_0 - |P_n| \tag{7-41}$$

Table IX shows the number of EPPs modulo 2^n of degree at most 2^{n-1} for few selected small values of n. It should be noted that for even small values of n=8 the numbers of resulting EPPs are huge.

Table IX. Number of Distinct and Equivalent PPs [31].

Ring $\mathbb{Z}_{2^n}$	# distinct PPs	# EPPs
n=4	2^{13}	$2^{48.8}$
n=6	2^{29}	2^{321}
n=8	2^{47}	2^{17923}

Therefore, it seems useful to seek an upper bound for the degree d of all distinct permutation polynomials. The following upper bound on d can be derived by making use of the formula (7-41) for (N_0=$|P_n|$), resulting with

$$\tilde{d} = \left\lceil \frac{\log_2 |P_n| - n}{n-1} \right\rceil \tag{7-42}$$

where the degree $\tilde{d}$ is the upper bound of the degree of distinct PPs of size n. The values of n, P_n are known and $\lceil \cdot \rceil$ is the ceiling function. The formula in (7-42) represents a necessary design rule for selecting such distinct permutation polynomials.

Table X shows the relation between the cardinality of PPs and the corresponding highest degree d of non-equivalent PPs over $\mathbb{Z}_{2^n}$. If all practical applications require n >3 for $\mathbb{Z}_{2^n}$, and all proposed new SIQPFs classes have degree 2, then **all resulting PPs in any class are distinct**.

Table X. Upper Bound of Degree of Distinct PPs [31].

Ring $\mathbb{Z}_{2^n}$	Maximum degree $\tilde{d}$
n=4	3
n=5	4
n=6	5
n=7	5
n=8	6

In the following, the cardinality of each distinct class of the new SIQPFs is computed:

Corollary 7.13: For n >2, the cardinality of a class C1: $P(x) = ax \overset{\oplus}{+} bx^2$ over $\mathbb{Z}_{2^n}$ is:

$$|C_1| = \begin{cases} 2(2^{n/2}-1) & : \quad for\ n\ \ even \\ 2(2^{n+1/2}-1) & : \quad for\ \ n\ \ odd \end{cases} \tag{7-43}$$

Proof:

For n even, and from theorem 6.8, the following is true:

$$a = 0\underbrace{1\cdots1}_{n-1} \ \ or \ \underbrace{1\cdots1}_{n} \ \Rightarrow |a| = 2$$

and,

$$b = \begin{matrix} 1 & & 1 \\ 0 & \cdots & 0 \end{matrix} \underbrace{0\cdots0}_{n/2} \qquad \Rightarrow |b| = 2^{n/2}-1$$

This implies, $|C_1| = |a| \cdot |b| = 2(2^{n/2}-1)$. For n odd and applying similar steps, then $|C_1| = |a| \cdot |b| = 2(2^{n+1/2}-1)$. □

Table XI shows the corresponding cardinalities for all new SIQPFs classes. Where, the procedure of corollary 7.2 is repeatedly applied for each class. For a practical example of 64-bit arithmetic where n=64 ,i.e., $\mathbb{Z}_{2^{64}}$, the cardinalities of the permutation classes $C_1 = 2^{33}$, $C_2 = 2^{65}$, $C_3 = 2^{97}$ and $C_4 = 2^{129}$.

Table XI. Cardinality of All Resulting SIPFs [31].

Class of SIPFs in $\mathbb{Z}_{2^n}$	**Cardinality**	
	***n*: Even**	***n*: Odd**
C1: $P(x) = ax \overset{\oplus}{+} bx^2$	$\geq 2(2^{n/2}-1)$	$\geq 2(2^{n+1/2}-1)$
C2: $P(x) = a \overset{\oplus}{+} bx \overset{\oplus}{+} cx^2$	$\geq 2(2^{n/2}-1)^2$	$\geq 2(2^{n+1/2}-1)^2$
C3: $P(x) = ax \overset{\oplus}{+} b(x^2 \vee D)$	$\geq 2^{n+1}(2^{n/2}-1)$	$\geq 2^{n+1}(2^{n+1/2}-1)$
C4: $P(x) = a \overset{\oplus}{+} bx \overset{\oplus}{+} c(x^2 \vee D)$	$\geq 2^{n+1}(2^{n/2}-1)^2$	$\geq 2^{n+1}(2^{n+1/2}-1)^2$

Notice that the given cardinalities in table XI represent worst case bounds. The exact cardinalities seem to be difficult to evaluate as equal mapping may happen in different mappings constellations. Therefore, the smallest cardinality values are used when evaluating the resulting cipher performance and cardinalities.

7.3. Hardware Complexity of SIPFs Classes

To implement SQIPFs in the targeted FPGA platform, an optimal and effective implementation strategy proposes to attain the same number of LUTs and DFFs, i.e. the ratio $R_{LUT/DFF} = \frac{\#of\ LUTs}{\#of\ DFFs}$ should be close to 1. This can be inferred from the fact that when consuming an LUT, its corresponding DFF in the same logic cell cannot be used elsewhere as its input is used for the LUT [98].

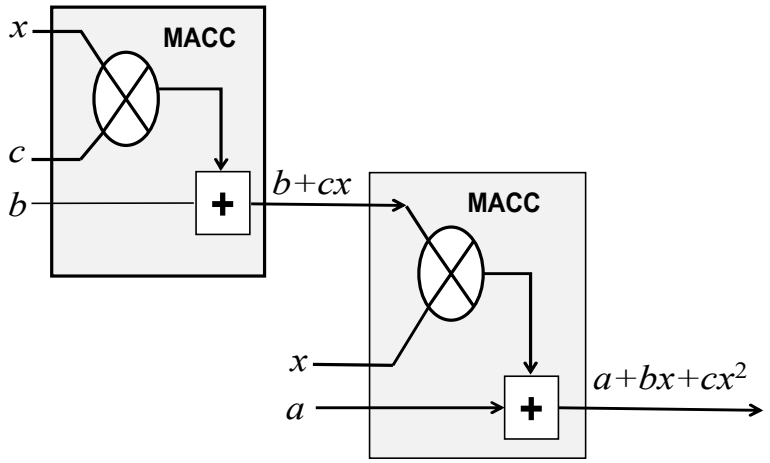


Figure 7-1. Implementing $a+bx+cx^2$ by Deploying Two MACCs [31].

These classes of SIQPFs can be implemented in both hardware and software or by combining HW/SW implementation scenario for the targeted SUCs [76]. In this case, the SIQPF can constitute one efficient class required for constructing the SUC cascade. The cryptographic strength of the generated permutations is attained through a huge number of possibilities of each SIQPF class simply controlled by the permutation function coefficients.

Figure 7-1 shows a basic hardware configuration for building the function $a+bx+cx^2$ for n=18 bits by using 2 MAACs. The designed SIQPFs are modeled in VHDL and synthesized to check their hardware complexity and performance. Mentor Graphics Modelsim ME package is used for simulation and simplify pro ME for synthesis.

By analyzing the FPGA resource usage for each function, a closed formula (7-44) was found for the number of MACCs, (N_{MACC}), LUTs (N_{4LUT}) and DFFs (N_{DFF}) for each class of SIQPFs and input data size in bits. Table XII shows the required hardware resources as a function of the number of n bits, where, U is a unit step function defined as

$$U(t-t_0)=\begin{cases}0:t<t_0\\1:t\geq t_0\end{cases} \tag{7-44}$$

Table XII. Hardware Complexity of All Resulting SIPFs [31].

$a+bx+cx^2$	$N_{MACC}=2U(n-3)+U(n-9)+6U(n-18)$
	$N_{4LUT}=U(n-1)+3U(n-2)+69U(n-3)+36U(n-9)+(2n+217)U(n-18)$
	$N_{DFF}=72U(n-3)+36U(n-9)+216U(n-18)$
$a+bx-cx^2$	$N_{MACC}=3U(n-3)+6U(n-18)$
	$N_{4LUT}=U(n-1)+3U(n-2)+109U(n-3)+4U(n-4)+(2n+217)U(n-18)$
	$N_{DFF}=108U(n-3)+216U(n-18)$
$bx+cx^2$	$N_{MACC}=2U(n-3)+U(n-9)+6U(n-18)$
	$N_{4LUT}=U(n-1)+2U(n-2)+70U(n-3)+36U(n-9)+(n+217)U(n-18)$

	$N_{DFF} = 72U(n-3) + 36U(n-9) + 216U(n-18)$
$bx - cx^2$	$N_{MACC} = 3U(n-3) + U(n-9) + 9U(n-18)$
	$N_{4LUT} = U(n-1) + 2U(n-2) + 109U(n-3) - 3U(n-4) + (n+217)U(n-18)$
	$N_{DFF} = 108U(n-3) + 216U(n-18)$
$a + bx \oplus cx^2$	$N_{MACC} = 3U(n-3) + 9U(n-18)$
	$N_{4LUT} = U(n-1) + 2U(n-2) + (n+109)U(n-3) + (n+199)U(n-18)$
	$N_{DFF} = 108U(n-3) + 216U(n-18)$
$bx \oplus cx^2$	$N_{MACC} = 3U(n-3) + 9U(n-18)$
	$N_{4LUT} = U(n-1) + 3U(n-2) + (n+105)U(n-3) + (n+216)U(n-18)$
	$N_{DFF} = 108U(n-3) + 216U(n-18)$

Table XIII shows the hardware complexity of the functions for some selected number of bits n up to 32 bits. From this it can be taken that $R_{LUT/DFF}$ of class (bx-cx^2) , for instance, is equal to 1.01, 1.009, and 1.1, when n =8,17, and 32, respectively.

Table XIII. Sample Hardware Complexity [31].

SIPFs	**Total Cost**			
	n=8	***n=17***	***n=32***	***$R_{LUT/DFF}$, n=32***
$a + bx + cx^2$	N_{MACC} 2	3	9	1.2
	N_{4LUT} 73	109	390	
	N_{DFF} 72	108	324	
$a + bx - cx^2$	N_{MACC} 3	3	9	1.2
	N_{4LUT} 109	109	390	
	N_{DFF} 108	108	324	
$bx + cx^2$	N_{MACC} 2	3	9	1.1
	N_{4LUT} 73	109	357	
	N_{DFF} 72	108	324	
$bx - cx^2$	N_{MACC} 3	3	9	1.1
	N_{4LUT} 73	109	358	
	N_{DFF} 72	108	324	
$a + bx \oplus cx^2$	N_{MACC} 3	3	9	1.1
	N_{4LUT} 117	126	357	
	N_{DFF} 108	108	324	
$bx \oplus cx^2$	N_{MACC} 3	3	9	1.2
	N_{4LUT} 117	126	389	
	N_{DFF} 108	108	324	

Notice also that, SIQPF $a + bx + cx^2$ exhibits the highest efficiency as it makes maximum use out of the same deployed number of MACCs.

To evaluate the software complexity, SmartFusion®2 SoC FPGA incorporates ARM Cortex-M3 that supports the Thumb2 instruction set. This set contains enhanced instructions as single cycle multiplication between two numbers of 32 bits.

Table XIV shows the time and memory implementation complexities of the same set of

permutation functions when using ARM Cortex software environment for some chosen numbers of bits *n*.

Table XIV. Software Performance and Complexity [31].

SIPFs	Total Cost	
	***n*=16**	***n*=32**
$bx+cx^2$	36 Bytes 15 cycles	28Bytes 13 cycles
$bx-cx^2$	36 B 15 cycles	28Bytes 13 cycles
$bx\oplus cx^2$	84 Bytes 29 cycles	36 bytes 16 cycles
$a+bx+cx^2$	60 bytes 22 cycles	36 bytes 18 cycles
$a+bx-cx^2$	76 bytes 27 cycles	36 bytes 18 cycles
$a+bx\oplus cx^2$	100 Bytes 34 cycles	44 bytes 19 cycles

7.4. SUC Cipher Construction and Security Analysis

In [147], the permutation T-function *f* used to construct unusual permutations by XORing of any pair of *f*(*x*), *x* as $f(x)\oplus x$. Following this work, the resulting SIQPF's classes can be extended by using this technique, for example, with $P(x)=a+bx\oplus cx^2$ as a SIQPF, the XOR of any pairs of *P*(*x*), *x* is a permutation $P(x)\oplus x$ but not necessarily a SIQPF. One of the most important results based on this discussion is in deploying them to counteract the weakness stated in theorem 7.3, 7.4, 7.5 and 7.6. For example, let $P(x)=3x+2x^2$ be a SIQPF in $\mathbb{Z}_{2^3}$ and let $G(x)=x\oplus 3$ be an XOR mapping where the bitwise XORing operation with any given value represents an involution. Therefore, the resulting function composition $f(x)=(3x+2x^2)\oplus 3$ is a SIQPF without fixed points. However, the previous discussion proves that XORing a SIQPF with the key *k* results with a round function $P(x)\oplus k$ for a block cipher which avoids fixed points, where *k* is a round key.

Assume that the proposed SUC follows a key-alternating cipher structure. Such a cipher consists of SIQPFs as round functions interleaved with XORing secret round keys to the state shown in Figure 7-2. In addition, the selected SIQPFs are completely unknown to anybody (as they are defined by a TNRG). In this case, the whole mapping is known as key-alternating Cipher *y* which is defined as follows [17]:

$$y=k_r\oplus P_r(\cdots P_2(P_1(x\oplus k_0)\oplus k_1)\cdots) \tag{7-45}$$

with key space $\{0,1\}^{(r+1)n}$ and $r>1$.

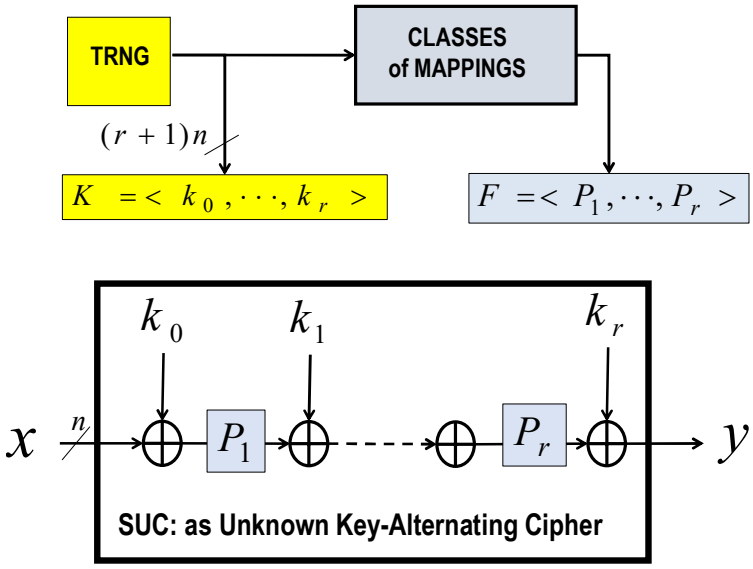


Figure 7-2. Key-alternating Format of the Targeted GENIE Created SUCs [31].

The main advantage of deployed involutive cascaded mappings (as SIQPFs) is that the same mapping cascade can be used for both encryption and decryption operations by just reversing the sequence of the round keys. The cardinality of the resulting cipher when using the key-alternating cipher structure of Figure 7-2. The cipher has *r* randomly selected self-inverse permutations $P_1,...,P_r$ and $(r+1)$ randomly selected *n*-bit keys.

The usable self-inverse permutations are included in the above 4 different classes $C_1, C_2, C_3,$ and C_4 from which $P_1,...,P_r$ can be randomly selected. Notice that, for highest security and to avoid fixed points in the total mapping, at least one permutation needs to be selected from the classes C_3 and/or C_4. We select few possible random cipher selection strategies to evaluate the cardinality of all possible selectable SUCs.

In this case, we consider the selectable mappings, in Figure 7-2, to be out of the set of all 4 classes $C_1, C_2, C_3,$ and C_4. Hence:

- The cardinality *S* of all classes of mappings is then:

$$S = |C_1| + |C_2| + |C_3| + |C_4| = (2^{n+1/2} - 1)\left(2^{\frac{n+1}{2}+1} + 2^{\frac{(n+1)^2}{2}}\right)$$

- As each cipher utilizes $(r+1)$ keys of size *n*, the cardinality of keys is: $2^{n\times(r+1)}$.
- The GENIE selects *r*-mappings randomly from *S*, hence there is *r*! possible placements of the mappings to build each cipher.

The GENIE selects randomly *r* mappings from the classes of mappings *S* and $(r+1)\times n$ key bits. The placement of the selected mappings is totally random. We investigate the following two selection cases.

Case 1.1: If the selection is done with allowed *Pi* repetition (i.e. the GENIE can select the same mapping multiple times for different *Pi s*), then the total number σ_{11} of all possible different key-alternating ciphers-selections including keys according to Figure 8 as an SUC having r rounds with an input size of n-bit is:

$$\sigma_{11} = r! \times S^r + 2^{n\times(r+1)}$$

Case 1.2: If the selection is done without *Pi* repetition, then the total SUC selections cardinality σ_{12} in that case according to Figure 7-2 is:

$$\sigma_{12} = r! \times \binom{S}{r} + 2^{n\times(r+1)}$$

As described in the SUC concept in Chapter 3, the cipher, generated by the GENIE, is not known to anybody.

7.4.1. Interpolation Attack

A well-known interpolation attack would successfully reveal a mapping *y* equivalent to the whole SIQPF cascade if, and only if, the SUC is given as a cascade of only algebraic classes of SIQPF such as $P(x) = a \pm bx \pm cx^2$. In this case, an adversary can compute such equivalent mapping *y* to all *r* rounds by just 2*r*+1 known plaintext/ciphertext pairs [152] [153] as follows,

$$y = f(x) = \sum_{i=1}^{r} \prod_{1\le j\le r, i\ne j} \frac{x - x_j}{x_i - x_j} \tag{7-46}$$

Note that an adversary needs just to know that the functions are quadratic ones and guess the number of rounds *r*. $f(x) = x + (x^2 \vee C)$ $P(x) = a \overset{\oplus}{+} bx \overset{\oplus}{+} c(x^2 \vee D)$

In another case, the interpolation attack is not applicable $2^{\frac{r}{r+1}n}$, when the selection of SIQPFs is drawn from non- algebraic classes. However, Klimov *et al.*[137] presented an attack scenario on T-function , when presented as a substitute for LFSR in a stream cipher. This attack works, for example, only if the size of *C* is small, such as *n*/3. In this case it would require $O(2^{n/3})$. Fortunately, there is no known attack on a T-function as a round function in a block cipher.

7.4.2. Distinguishing Attack

In [17], Bogdanov *et al.* conjectured that the query complexity of the distinguishing attack on a key-alternating cipher is , where, *n* is the cipher input size, and *r* is the rounds number. In [154], Steinberger presented an improved security bound $2^{\frac{3}{4}n}$ for the distinguishing attack on a key-alternating cipher, when $r > 2$. In [155], Lampe *et al.* showed that if *r* is even, the security bound attains $2^{\frac{r}{r+2}n}$. In [156], the tight security bound $2^{\frac{r}{r+1}n}$ of the distinguishing attack on a key-alternating cipher was proved, i.e., no additional security bound can be attained.

It is therefore conjectured, that the security bound $2^{(\varepsilon)n}$ is enhanced by a factor equal to the product of the cardinalities of the deployed SIQPFs, where ε is a function of *r*. Ongoing research is conducted to answer this still open question. However, the proposed SUC cipher design fulfils at least the state-of-the-art security requirements for standard good ciphers.

Algorithm .2

1 Enter n=32, cipher rounds r =8, and select randomly one SIQPF P.

2 Select randomly 10000 input values $x^{(i)}$, where i=1, ... ,10000.

3 Determine the dependence matrix A;

$$A_{ji}(P) = 10^{-4} \sum_{i=1}^{10000} P(x^{(i)}) \oplus P(x^{(i)} \oplus e_j)$$

Where, $e_j = (\delta_{j1}, \delta_{j2}, \cdots, \delta_{j32})$, δ_{ji} is a Kronecker's delta.

4 Return the average of A.

7.4.3. Statistical Properties of the Resulting SUC

In the following section some statistical properties of SIQPFs are investigated such as diffusion [20] and frequency prediction [37]. Here, the statistical properties provide an initial proof of the indistinguishability of the proposed SUCs.

A. Diffusion Properties

The essential definition of a diffusion is to determine the number of changed output bits when one input bit has been changed. Ideally, the changing ratio of the output bits is 50%. However, a T-function is defined as a mapping in which bit i of the output depends on 0, 1, …, i bits of the inputs [126]. Thus, this indicates that changing the first input bit affects all n output bits, changing the second input bit affects the last n-1 output bits, etc. changing the last input bit, therefore, affects the last output bit.

To test this property, the hamming distance between outputs of randomly selected SIPQF by changing one input bit every time is measured. The applicable algorithm.2 is defined as a simulator to determine the amount of diffusion.

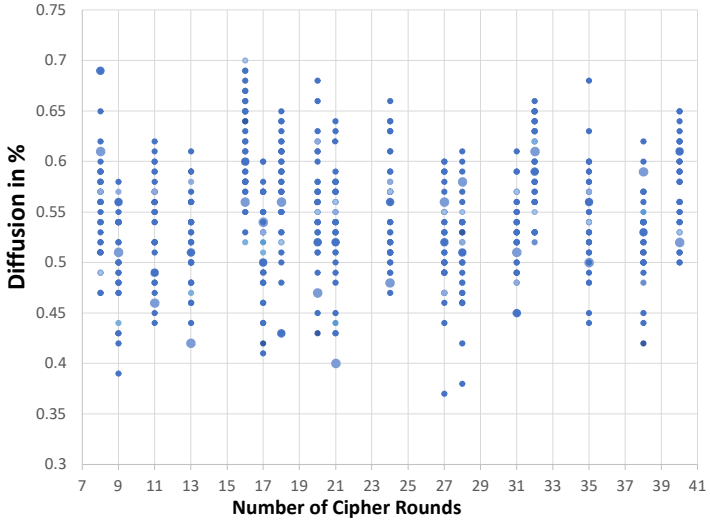


Figure 7-3. SIPQF Classes Diffusion by using 1000 Random Pairs [31].

The results in Figure 7-3 show that increasing the number of rounds in (7-45) doesn't

change the statistical distribution of the diffusion of any resulting cipher in the class. In this case, the simulation indicates that the average of diffusion is close to 50% for repeatedly using a single SIPQF with *r* (iterations) rounds.

B. Ciphertext bit Frequency Distribution

A probability distribution $P[y_i=1]$ indicates the level of predictability of the bit output y_i, when y_i =1. The ideal case corresponds to $P[y_i=1]$ =50%.

To test this, 10000 random input/output pairs are used for selected SIPQF, where, the predication of each output bit y_i is given based on statistical distribution of $P[y_i=1]$. This procedure has been performed and repeated for 10 randomly selected SIPQF. Results in Figure 7-4 shows a high unpredictability of the bit output y_i, while $P[y_i=1]$ is close to 50%.

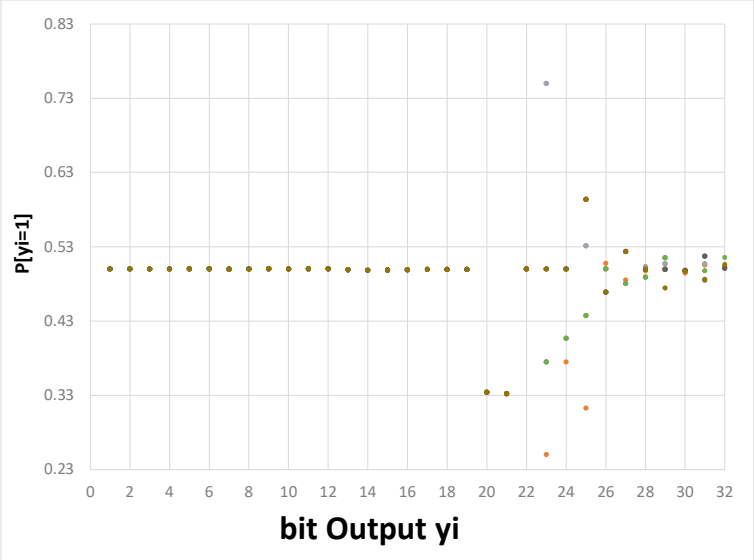


Figure 7-4. The Probability that Any Output Bit is 1 [31].

7.5. Summary

New classes of low-complexity self-inverse permutations based on T-Functions (SIPFs) were presented. The target usage of such functions is for the creation of so-called Secret Unknown Ciphers (SUC) at very low cost to serve as clone resistant digital PUFs. The permutations algebra deploys the optimized usage of multiplier/Mathblocks units which are often not consumed in FPGA applications. As a result, high quality ciphers may be embedded with negligible cost when using unconsumed resources in such SoC units. The resulting new cipher classes based on arithmetic is very promising in their security and quality. The resulting SUC structures to be used as PUF alternatives are very attractive as they are efficiently usable in emerging future smart application scenarios.

PART III

"*None of us knows what might happen even the next minute, yet still we go forward. Because we trust*."

Brida

Paulo Coelho (1947-)

8. APPLICATION PROTOCOLS FOR SUC-BASED IDENTITY

Trust Relationship (TR), Identity Trust (IT), and Trusted Communication Link (TCL) play a crucial role in the complex networks from a security point of view. Trust is not only security, it also relates to many other factors such as privacy, reliability, availability, ability, etc. [157]. Therefore, trust management plays a crucial role within interconnected networks and their applications such as IoT [158], vehicle communication [159], etc. These applications include the communication between smart devices within, for example, smart grids, intelligent transportation systems, smart cities, and industrial automation, etc. [160]. One of the most important challenges facing such applications is the large number of devices connected to the internet. This exceeded by far the number of users between 2008 and 2009. In fact, "it is estimated that by the year 2020 this number will reach 50.1 billion [161]" and increase significantly in the future [162]. The question then arises, "How to improve the communication security in large complex networks?".

In this chapter, SUCs are proposed as a useful tool for creating a TR between devices in complex open communication networks. as well as being deployed to enhance the security level of such networks against cloning and manipulation attacks [163]. These goals can be achieved by:

- Deploying SUCs in each device.
- Designing generic protocols using SUCs to build a trust chain in the networks.
- Utilizing the designed generic protocols to build an authenticated network.

The contents of this chapter have been published in [163], [30] and [164].

8.1. Introduction

Currently, complex topology and/or hub structure are considered the common properties of the networks. A reliance on hubs in a network has a serious drawback: If an adversary can successfully attack or disrupt a few nodes-hubs of the network, then the network will divide into small groups of isolated nodes [163]. Therefore, improving security level of the networks is one of important challenges in IoT, etc. For instance, several studies investigated and discussed the security level of a network such IoT networks [165] [166], and determined the threat models in the networks [167].

In the following, some fundamental security requirements of complex networks are reviewed. The goal is to prepare the grounds for utilizing SUCs to enhance the reliability and security of such networks.

Networks need to be secure and immune to different threats such as illegitimate use, information leakage, integrity violation, Denial of Service (DoS), etc. With this vast amount of threats, it is challenging to ensure a high security level of a network. The five following key security requirements should be taken into consideration for every communication network:

- **Integrity.** This refers to the ability to verify if the transmitted data has been tampered with or changed via the network. Integrity is required to ensure the reliability of transmitted data [168].
- **Confidentiality.** This refers to the prevention of sensitive and private data from reaching the unauthorized parties so that the transmitted data via the network remains confidential [168].
- **Availability.** This refers to the legitimate parties' ability to have access to the resources if they need it.
- **Access control.** This refers to keeping the unauthorized parties out of the network.
- **Authentication.** This refers to the identification of an individual component via the network based on identity or password, etc.

Within the scope of this chapter a new identification mechanism based on SUCs for building trusted link via a network is presented. Essentially, the security requirements of identification mechanisms for IoT were presented in [165]. In [169], Sarma and Girão proposed that a subset of the user data can serve as a virtual identity. The data subset could be gathered relating to many services and network parameters such as mobile numbers, software identities, vehicle identity, etc. A similar approach was introduced in [170] by combining modern cryptographic techniques such as the ElGamal cryptosystem with a virtual identity. In [171], Vogt built a trusted link between two parties by using the public-private key technique in a Small-world network [172]. This approach exhibits a specific network topology which does not contain nodes-hubs.

It has been shown that the IoT applications and cyber-physical systems couldn't be completely protected with pure software solutions [87]. Therefore, hardware security solutions are proposed to ensure securing IoT devices [173]. For instance, PUF as a hardware scheme provides each hardware device with a unique signature. Furthermore, although the recent proposed PUFs are considered one of the promising hardware solutions, they cannot fulfill lightweight requirements [10]. Here, SUCs as a combination of software and hardware specifically offer an optimal balance between security and lightweight design [163]. Therefore, several applications deploying SUC technique have been published recently in [174], [175], and [176]. For instance, a low-cost two-way protocol deploying SUC for a smart meter system was proposed in [164]. This proposal has exhibited a promising long term resilience and stable security architecture.

In the following section, a SUC-system model is defined. The operation steps of the SUC-system are illustrated carefully. Furthermore, a user credential is deployed in the SUC-system as a proof of the ownership of the proposed physical identity generated by a specific SUC.

8.2. SUC- System Set Up Model

In [177], Vaudenay presented a security model of the Radio-frequency identification (RFID) system. Following this work, the framework of a system network using SUCs is modelled after the Vaudenay-Model without being restricted to the RFID system [177]. Such a network was considered in [30] as a generic system which consists of a single Trusted Authority (operator-TA), a single data-base (DB) operated by server S and N devices. In this case, SUC- system is defined as follows:

Definition 8.1 [30]**:** A SUC-System is a tuple of procedures and protocols (S, DB, ServerSetup, DeviceSetup, Ident, Update) that are defined based on a security parameter $l \in \mathbb{Z}$ as follows:

- **ServerSetup(1^l)**: A procedure carried out by TA. It starts with generating ID_i as an open identity for the device. After that, TA assigns, for example, ID_A to SUC_A representing a device A. The TA then initializes a DB from server S.
- **DevicePersonalization(A, ID_A)**: A procedure carried out by TA. It starts with sending challenges as plaintexts $\{x_1, ..., x_T\}$ to the device, e.g. A as in Figure 8-1, and storing the corresponding ciphertexts $\{y_1, ..., y_T\}$, where $SUC_A(x_i)=y_i$; i=1, …,T, in a secret record in the DB of S.
- **Ident ()**: A generic protocol between a device and S uses a plaintext/ciphertext pair from a device record in the DB for the identification process. Note that S selects randomly a plaintext/ciphertext pair from a device record in DB.
- **Update ():** A generic protocol between a device and S. It starts when S picks the final pair from the device record. Then, the device starts updating its record by creating a new response list $\{y'_1, ..., y'_T\}$.

Figure 8-1 illustrates how SUC-system is initialized in the enrollment/personalization phase as follows [30]:

1. TA assigns *N* open identities IDs to *N* devices with *N* different embedded SUCs as serial numbers for each device. Here, a node *A* denotes a device *A* with an embedded certified SUC_A and an open identifier ID_A. Note that no clear relation between *A* and ID_A and the corresponding embedded SUC_A exists.
2. TA sends a set of plaintexts $\{x_1, ..., x_T\}$ to the device, e.g. *A*.
3. TA stores the corresponding ciphertexts $\{y_1, ..., y_T\}$, where $SUC_A(x_i)=y_i$; $i=1, \ldots,T$, in a secret record in the DB of *S*.

After that TA distributes N certified SUCs to anonymous users. The SUC owners can check if the acquired SUC is authentic or not with a help of the TA.

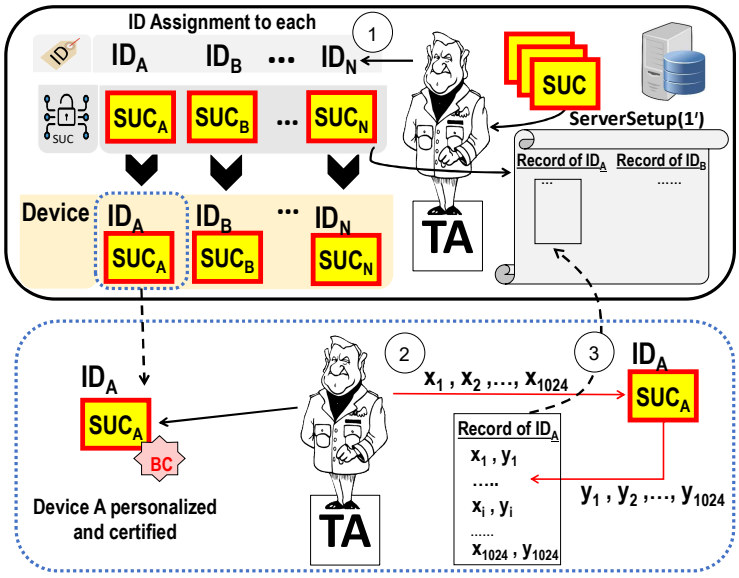


Figure 8-1. SUC-System Model: ServerSetup and DevicePersonalization [30].

In the following sections, more details about the SUC-system are presented as well as some extra features are discussed such as a user credential and CRPs management.

8.2.1. User Credentials

A user credential consists of a username, identity, public information etc. which is related to a particular user [163]. In other words, the credential is deployed to prove an identity to a system [178]. In this proposal, a credential is used to prove the ownership of the proposed physical identity generated by a specific SUC.

The device-credential is composed of several information as depicted in Figure 8-2: ID_A, as an open identifier, a public proof value Y_A (a public credential), start/end-dates, and trusted relations. The generation of a public credential Y_A requires the ownership of the device *A* and its SUC_A, where, X_A is a randomly chosen cleartext and $Y_A=SUC_A(X_A)$. Furthermore, Y_A is a one-time public key to be used for authentication and will be changed for every authentication process. The only entity which can decrypt Y_A is the owner of the device *A* by using its SUC_A [163].

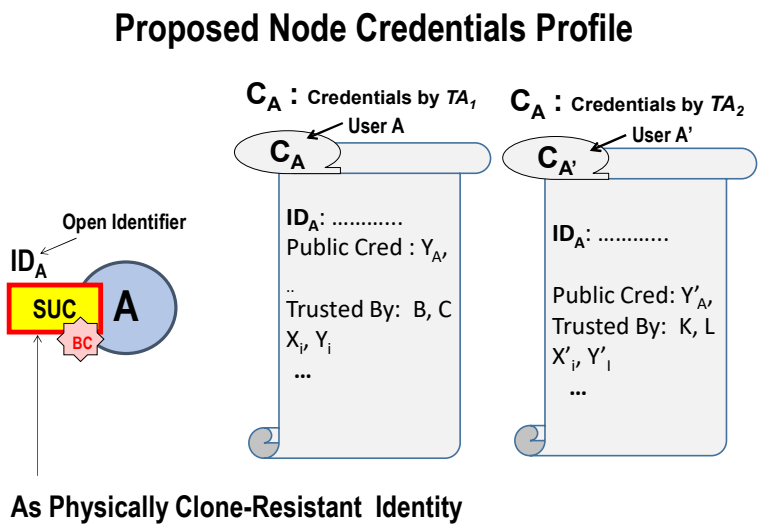


Figure 8-2. The proposed Credentials Profile [163].

Figure 8-2 shows that a device with an embedded SUC_A can be used by two or more TAs such as TA_1 and TA_2. Here, TA_1 and TA_2 can define two credentials independently for two users A and A'. TA_1 and TA_2 can use own secret input keys for the SUC to isolate the applications.

It should be clear that the authentication protocol ensures the secure verification of the identity while the identification protocol is limited to an unverified claim. In this chapter, the protocols provide the device authentication and therefore also the device identification. However, identification prior to authentication is more practical as presented in chapter 4.

8.2.2. CRPs Management: PUF vs SUC

Another significant advantage of SUC is when all stored SUC-CRPs are consumed. The invertibility of the SUC allows linear scalable monitoring and marking all used pairs. Such a feature is still not available in conventional PUFs. In this section, a comparison between PUF and SUC methods for CRPs management are presented. The comparison covers the required memory complexity of storing CRPs in a DB, and the efficiency of the update protocol.

A. CRPs Storage Complexity

In the case of PUF: when all stored CRPs are consumed, the PUF records need to be updated in a secure environment [46]. A dynamic update is also possible, a new CRP is generated after every use and encrypted by the previous used pair [179] [46]. In most solid applications, the PUF needs to identify all used pairs. If the PUF's input size is n and the number of used pairs is T, then the required store size (memory complexity) MC_{PUF} is:

$$MC_{PUF} = n \cdot T \quad Bits \tag{8-1}$$

As T unpredictable randomly selected challenges need to be stored. The PUF needs to check the whole stored list of T-random challenges to find if a challenge was already used. This complex search is not necessary in case of SUC.

In the case of a SUC: the TA chooses a secret displacement of S_0 (see Figure 8-3) and

uses S_0 to define a very short secret list of 2^r cleartexts $x_{i_t} = S_0||t$, where, and $t = 0,\cdots,2^r - 1$. Then, the TA just encrypts x_{i_0} to $x_{i_{2^r-1}}$ as cleartexts to get the corresponding ciphertexts y_0 to y_{2^r-1} in a secured environment. After that, the TA stores the resulting ciphertexts y_0 to y_{2^r-1} in a record of SUC for this device.

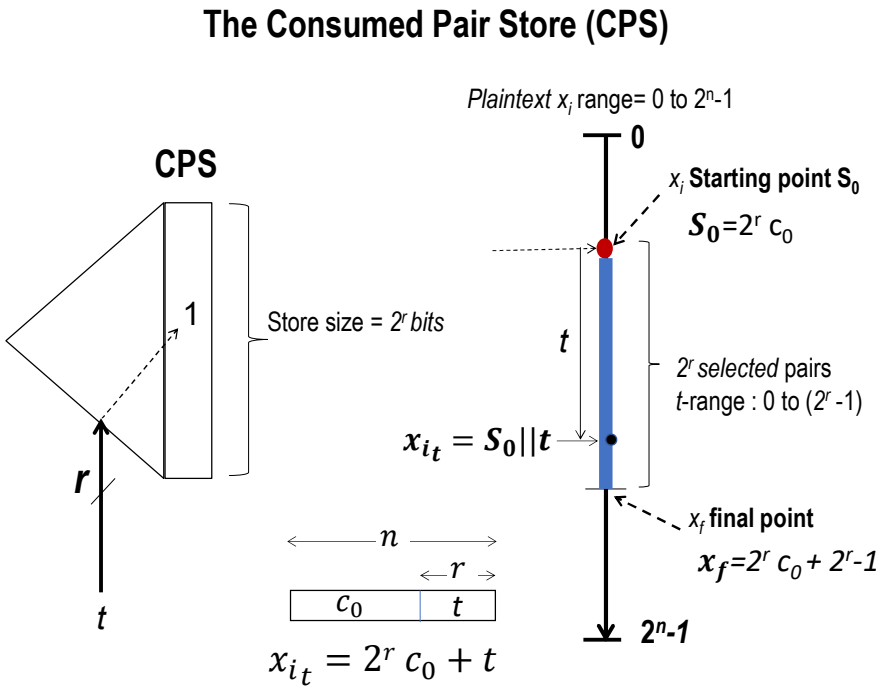


Figure 8-3. The Consumed Pair Store to Monitor the used Pairs.

On the device side, a Consumed Pair Store (CPS) of the SUC monitors the used (x_i,y_i) pairs as shown in Figure 8-3. The CPS address size is r bits; so that, the CPS memory size is 2^r bits. The target of deploying CPS is to prohibit pair multi usage. For example, if a device needs to use x_{i_t} in a transaction, where x_{i_t} is computed as $x_{i_t} = S_0||t$, and $t = 0,\cdots,2^r - 1$. Then, the device can mark only the t in CPS to prohibit multi usage of x_{i_t}. Therefore, the store size of the used pairs in the SUC-device is MC_{SUC}:

$$MC_{SUC} = \underbrace{CPS}_{Memory\ Size} + \underbrace{Secret\ Displacements}_{Memory\ Size} = 2^r + (n-r)T / 2^r \quad Bits \quad (8\text{-}2)$$

The CPS allows linear scalable monitoring and marking all used (x_i,y_i) pairs, which is only possible if the CRPs are cleartext-ciphertext pairs. This mechanism is not applicable in traditional PUF as revealing one challenge clear text allows finding all other challenges. Therefore, only randomly selected challenges are usable without any structure and each one need to be stored in full length as n-bits.The ratio between MC_{PUF} and MC_{SUC} is given as follows:

$$\frac{MC_{SUC}}{MC_{PUF}} = \frac{2^r}{nT} + \frac{(n-r)}{n \cdot 2^r} \quad (8\text{-}3)$$

We assuming that the whole T CR-pairs are completely consumed. Here, we distinguish

between two cases: First case, if T is a small number such as $T=2^r$, that corresponds to the usual operation in the most applications. Then, the ratio between MC_{PUF} and MC_{SUC} is,

$$\frac{MC_{SUC}}{MC_{PUF}} = \frac{2^r}{nT} + \frac{(n-r)}{n \cdot 2^r} \approx \frac{1}{n} \tag{8-4}$$

Table XV shows that the SUC-based pairs monitor requires much less memory store (reduced by the factor $\approx 1/n$) for practical ranges compared to that of a PUF-based monitor.

Table XV. SUC vs PUF Used-Pairs Memory Complexity, when $T=2^r$.

n	***T***	***MC*$_{PUF}$**	***MC*$_{SUC}$, *r* =10**	***MC*$_{SUC}$/*MC*$_{PUF}$**
64	2^{10}	2^{16}	$\approx 2^{10.07}$	$\approx 2^{-6} \approx 1/64$
128	2^{10}	2^{17}	$\approx 2^{10.1}$	$\approx 2^{-6.9} \approx 1/128$

Second case, if T is a large number, that corresponds to a few applications. Then, the ratio between MC_{PUF} and MC_{SUC} is,

$$\frac{MC_{SUC}}{MC_{PUF}} = \frac{2^r}{nT} + \frac{(n-r)}{n \cdot 2^r} \approx \frac{(n-r)}{n \cdot 2^r} \tag{8-5}$$

Table XVI shows that the SUC-based pairs monitor requires much less memory store (reduced by the approximately factor $\approx 2^{-10}$) compared to that of a PUF-based monitor.

Table XVI. SUC vs PUF Used-Pairs Memory Complexity, when T is a Large Number.

n	***T***	***MC*$_{PUF}$**	***MC*$_{SUC}$, *r* =10**	***MC*$_{SUC}$/*MC*$_{PUF}$**
64	2^{16}	2^{22}	$\approx 2^{12.12}$	$2^{-9.87} \approx 2^{-10}$
	2^{20}	2^{26}	$\approx 2^{15.78}$	$2^{-10.21} \approx 2^{-10}$
128	2^{20}	2^{27}	$\approx 2^{16.89}$	$2^{-10.11} \approx 2^{-10}$
	2^{32}	2^{39}	$\approx 2^{28.6}$	$2^{-10.2} \approx 2^{-10}$

B. Update Consomed CR-Pairs

Assuming that r=10 and after marking all used x_i of the CRPs from 0 to 1023, the update protocol substitutes securely a new list of 1024 (x_i,y_i) pairs.

Figure 8-4 illustrates the update protocol for SUC with input size n=128 and r=10 as follows:

1. The TA assigns the last pair (x_i,y_i) as (x_{final},y_{final}) from a secret record of device A in DB.
2. The TA chooses a random value a.
3. The TA sends $y_{final} \| a$ to device A.
4. Device A computes a new displacement S_1 as $S_1 = F_{x_{final}}(y_{final}, a)$.

5. Device *A* generates a new list of 2^r pairs as $y_t = SUC_A(S_1||t)$ for *t*=0 to 1023, where, $x_t = S_1||t$ for all *t*.
6. The resulting list is sent encrypted by using x_{final} as the key to the TA: $F_{x_{final}}(y'_0|| \cdots ||y'_{2^r-1})$
7. The TA computes $F_{x_{final}}(y_{final}, a)$ and sets it as a new displacement S_1.
8. The TA decrypts the list and updates its (x_i,y_i) pair's list.

Now, device *A* resets its CPS memory and the system is ready to monitor the new 2^r pairs.

The proposed CPS exhibits and offers an alternative and effective solution to the problem of storing and monitoring all generated CRPs in case of PUFs. (see 3.4 Problem Motivating the research work- CRP Management and Storage Problem).

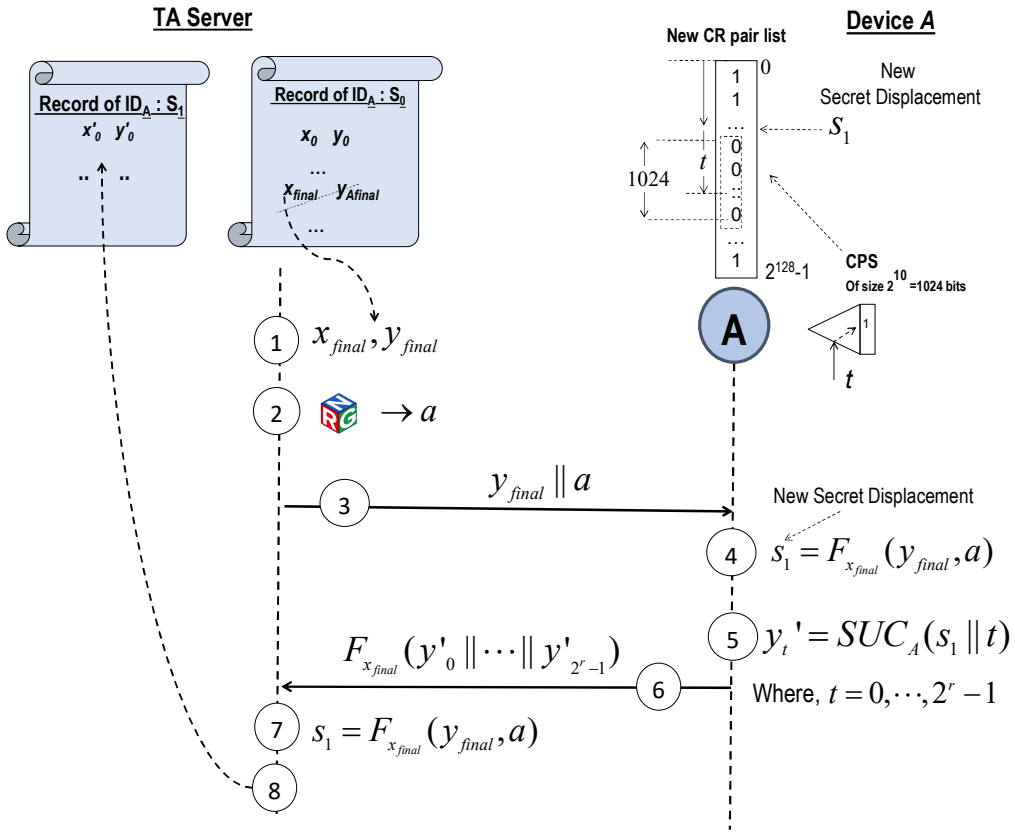


Fig. 8-4. Update Protocol for of Device's Secret Record Every 1024 Reading. Adapted from [30].

Now the SUC-system is ready to use. In the following section, efficient TR between devices in a large complex network is presented. All proposed protocols run over a fully insecure channel between some devices and a TA server. Impersonation attacks can also be applied at any time. The proposed protocols consist of the following major components: the devices with the embedded SUCs, the public credentials C_i, TA server, a standard cipher F_K with *K* as a secret key, TRNG, and a standard hash function *H*.

8.3. SUC Trust-Chaining in Large Complex Networks

In this section, few generic protocols are presented to build a TR between devices with embedded SUCs. The target of this section is to show an efficient trust-chaining in a large network when SUCs are involved. A hub node is deployed to be a mediator for building TR with a high security level of the resulting chain.

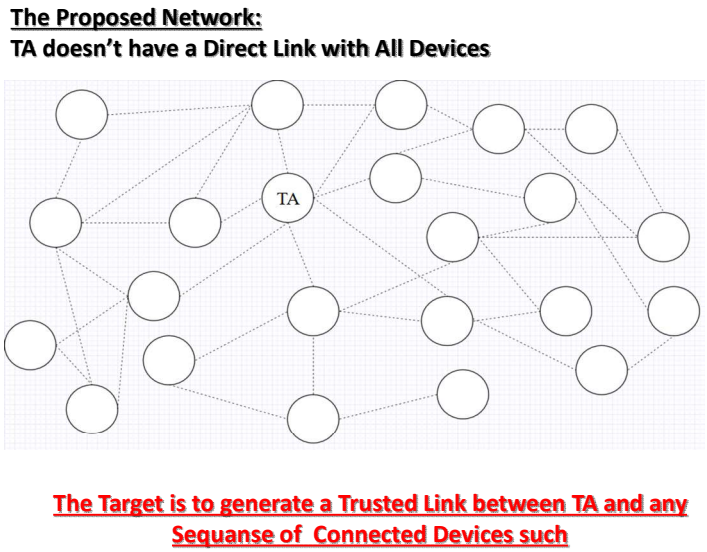


Figure 8-5. The Proposed Network with TA as a Central Hub.

In the following, the target is to create a trusted link between the TA and any sequence of the devices even there is not a direct link between TA and some device in the network. Figure 8-5 shows an example of the proposed network. Note that there is no direct link between TA and all devices. The desired trusted link can be created under the assumption that all devices in the network have their individual SUCs.

8.3.1. P_1: Generic Authentication Protocol for a Single Device

A possible generic identification protocol P_1 between a device A with embedded SUC_A and a TA is proposed as shown in Figure 8-6.

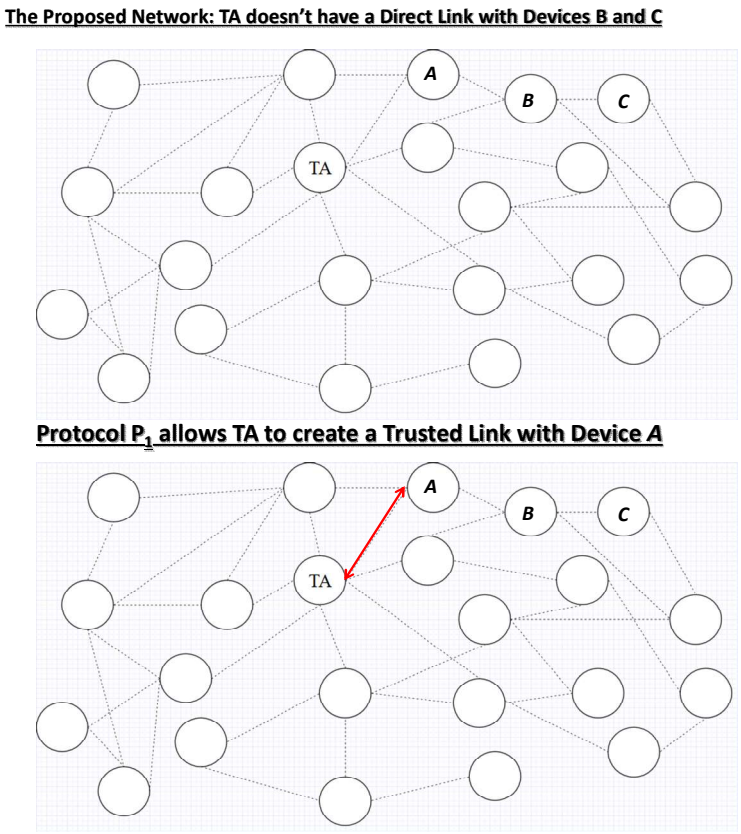


Figure 8-6. A Network with TA as a Central Hub and other possible Mediator or Hub Device.

In the following protocol P_1 will be described in detail explaining how a trusted link between the TA and device A can be securely created allowing a precise identification. Protocol P_1 can proceed as shown in Figure 8-7:

1. Device A randomly chooses X_{A_i} and uses SUC_A to compute Y_{A_i}.
2. Device A registers Y_{A_i} as public credentials in C_A and sends $C_A \| F_{X_{A_i}}[C_A]$ to TA, where $F_{X_{A_i}}$ is a standard cipher with X_{A_i} as a secret key.
3. TA finds the corresponding cleartext X_{A_i} for Y_{A_i} from A's secret record, and decrypts $F^{-1}_{X_{A_i}} F_{X_{A_i}}[C_A] = C'_A$
4. If $C_A = C'_A$, then device A is authentic.
5. TA marks the pair (X_{A_i}, Y_{A_i}) as used and never uses it again.

SUC_A/user A is now enrolled to the network and TA trusts device A. However, a shared session key between TA and device A is still required to ensure the communication between TA and device A. Particularly, TA and the device A should start the second phase of P_1 together to generate a shared session key Z_{A-TA}. Otherwise, the communication between TA and device A is not protected.

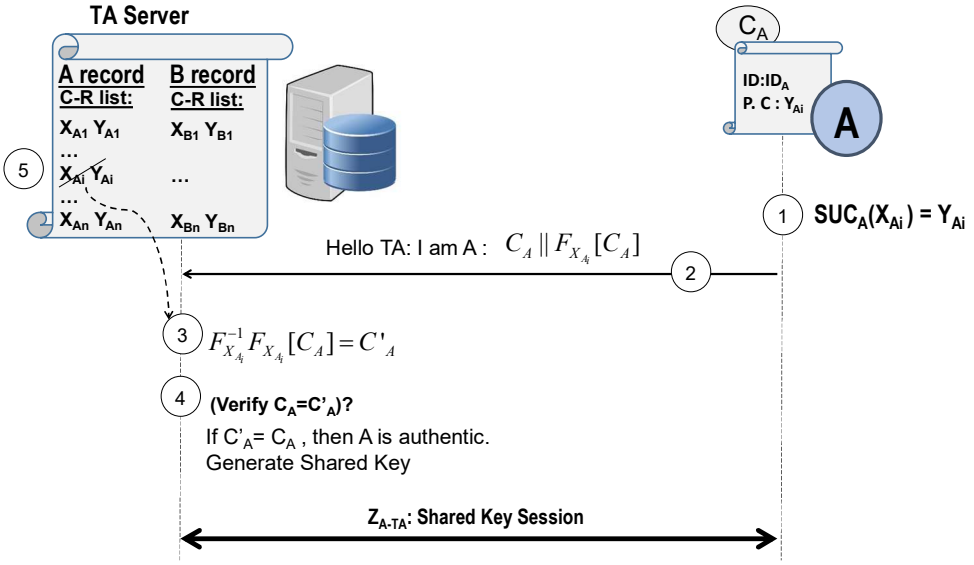


Figure 8-7. Generic Authentication Protocol for a Single Device. Adapted from [163].

The proposed protocol for generating a shared session key (SSK_1) can be described as shown in Figure 8-8:

1. TA server randomly selects a pair as ticket T_A=(X_{A_j} , Y_{A_j}) from *A*'s secret record.
2. TA randomly chooses a random value R_{TA}.
3. TA encrypts R_{TA} as $F_{X_{Aj}}[R_{TA}]$ and sends $F_{X_{A_j}}[R_{TA}] \| Y_{A_j}$ to device *A*.
4. Device *A* decrypts Y_{A_j} by its SUC_A to get $SUC^{-1}{}_A(Y_{A_j}) = X_{A_j}$.
5. Then device *A* decrypts $F_{X_{A_j}}[R_{TA}]$ by using X_{A_j} to get R_{TA}.
6. Device *A* randomly chooses R_A.
7. Device *A* sends $F_{X_{A_j}}[R_A]$ to TA.
8. TA decrypts $F_{X_{A_j}}[R_A]$ to get R_A.
9. Now, TA and the device *A* compute the session key $Z_{A\text{-}TA}$ as the hashed value:
$$Z_{A-TA} = H(R_A, R_{TA}, X_{Aj}) \qquad (8\text{-}6)$$
10. TA marks the pair (X_{A_j} , Y_{A_j}) as used and never uses it again.

The previous proposed protocol P_1 attains the following security functions:

- Device *A* with embedded SUC_A or user *A* is securely enrolled to the network.
- A shared session key $Z_{A\text{-}TA}$ created between TA and the device *A* to ensure the security of the communication between them.
- The trusted link between TA and device *A* cannot be changed or faked later.

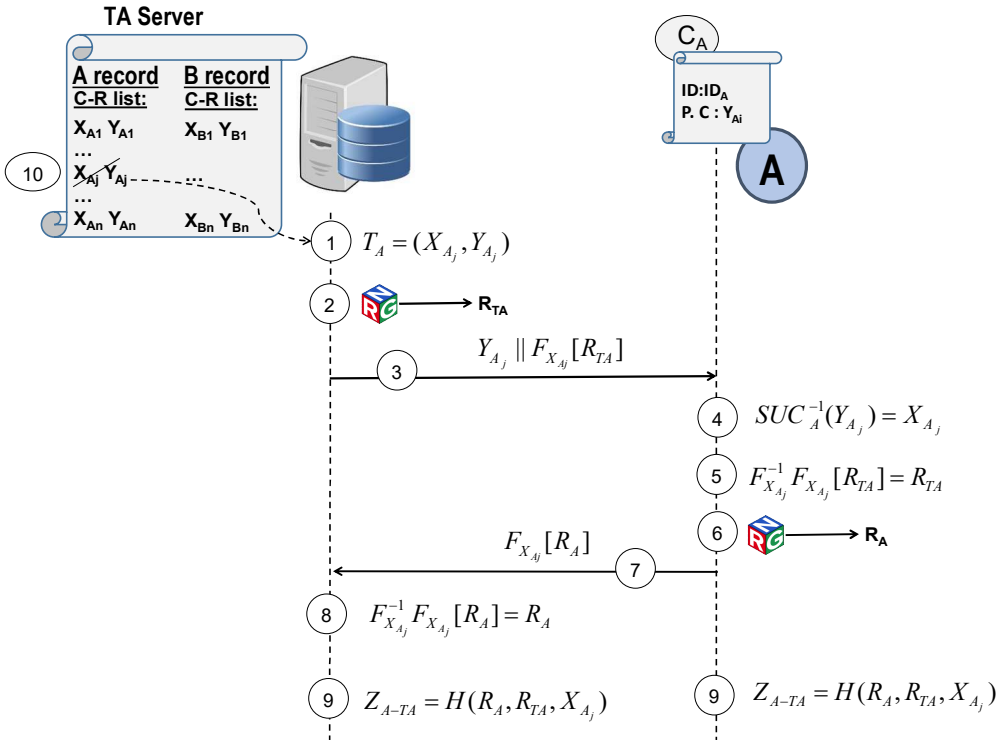


Figure 8-8. Shared Session Key Protocol between TA and Device *A* Using SUC. Adapted from [163].

8.3.2. P2: Authenticating a Device by One Mediator

Assume that device *A* has a trusted link with TA based on P_1 and device *B* has no direct link to TA. In this case, a trusted device *A* acts as a mediator (as a hub) between TA and device *B* as depicted in Figure 8-9.

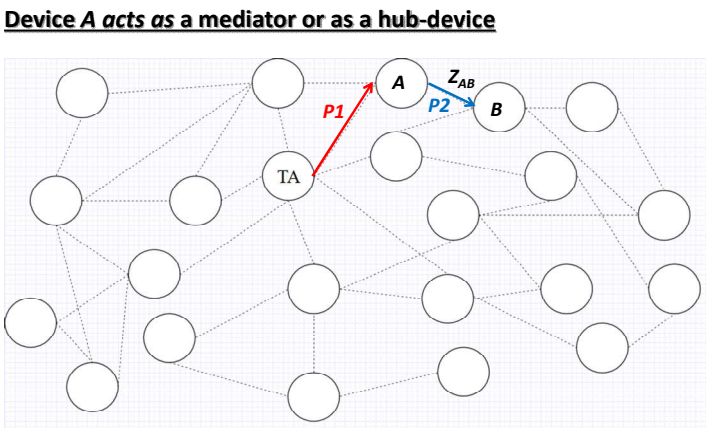


Figure 8-9. A Network with TA as a Central Hub and other possible Mediator or Hub Device such as *A*.

Figure 8-10 illustrates a possible protocol P_2 between a mediator such as *A* and any device such as *B*. This protocol generates a single trusted links as follows:

1. Device *B* randomly chooses X_{Bi} and computes Y_{Bi} by using its SUC_B.

2. Device B uses Y_{Bi} as credentials in C_B and sends $C_B \| F_{X_{Bi}}[C_B]$ to device A.
3. A trusted-device A forwards $C_B \| F_{X_{Bi}}[C_B]$ through its secured link to the TA.
4. TA finds the corresponding X_{Bi} value to Y_{Bi} from B's records, and then the TA computes $F^{-1}_{X_{Bi}}[C_B] = C'_B$ to verify if device B is authentic. This is the case when $C_B = C'_B$.
5. If device B is authentic, the TA selects a new pair T_B=(X_{Bj},Y_{Bj}) from B's records and sends it as an encrypted ticket of device B exclusively for A's use as $F_{XA_k}(T_B) \| Y_{A_k}$
6. A trusted-device A should then trust device B and use its SUC_A to compute X_{Ak}.
7. Then the trusted-device A decrypts the received ticket $F_{X_{A_k}}(T_B)$ to get T_B=(X_{Bj},Y_{Bj}) as follows:

$$F^{-1}_{X_{A_k}} F_{X_{A_k}}[T_B] = T_B = (X_{Bj}, Y_{Bj}) \tag{8-7}$$

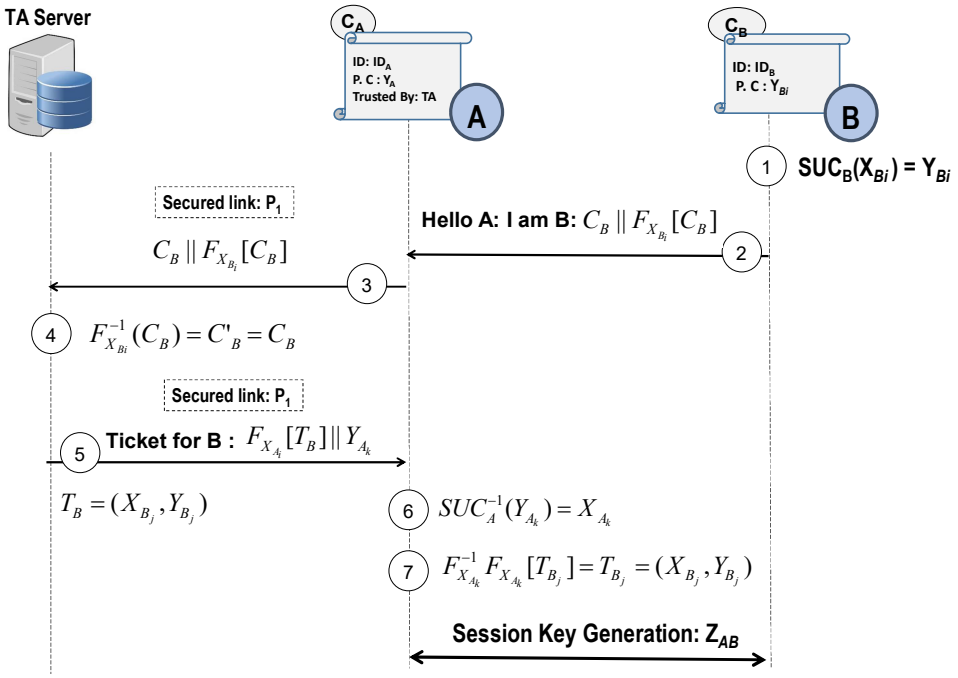


Figure 8-10. Authenticating a Device by one Mediator. Adapted from [163].

Now, the trusted-device A starts generating a shared session key, such as Z_{AB}, with device B by using the SSK protocol as shown in Figure 8-10. Note that a dashed rectangular denotes a secure link between two parties using an authenticated shared key for the communication. Furthermore, any two devices with the embedded SUCs can generate a session key that is constructed to ensure the security of the communication between these devices.

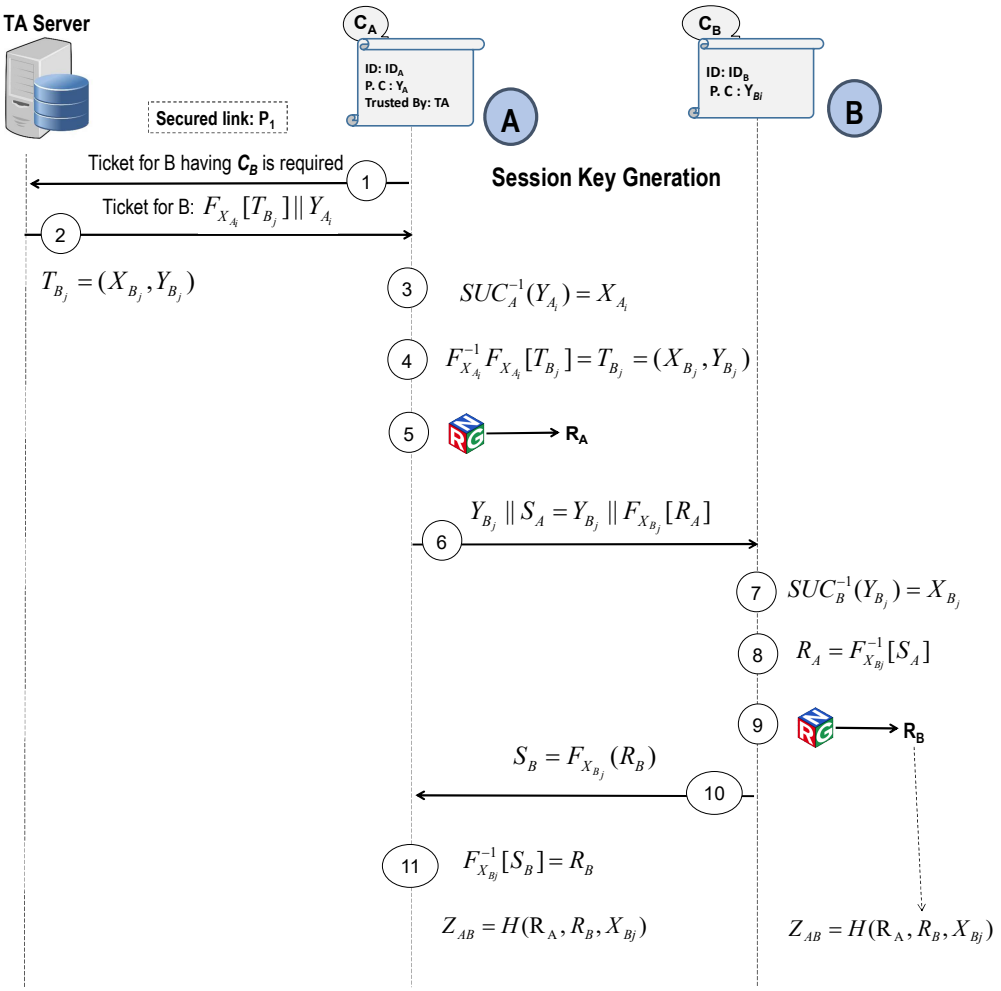


Figure 8-11. Shared Session Key Protocol between Two Devices with embedded SUCs. Adapted from [163].

The proposed protocol for generating shared session key (SSK_2) can be described as shown in Figure 8-11:

1. Device *A* sends the TA server a ticket request for device *B*.
2. TA server randomly selects a pair as ticket $T_{B_j} = (X_{B_j}, Y_{B_j})$ from *B*'s secret record. After that TA encrypts the chosen ticket T_{B_j} by using a pair from *B*'s secret record as $F_{X_{A_i}}[T_{B_j}]$ and sends $F_{X_{A_i}}[T_{B_j}] || Y_{A_i}$ to device *A*.
3. Device *A* decrypts Y_{A_i} by its SUC_A to get $SUC^{-1}{}_A(Y_{A_i}) = X_{A_i}$.
4. Then device *A* decrypts $F_{X_{A_i}}[T_{B_j}]$ by using X_{A_i} to get the ticket $T_{B_j} = (X_{B_j}, Y_{B_j})$.
5. Device *A* randomly chooses R_A.
6. Device *A* sends $S_A || Y_{B_j} = F_{X_{B_j}}(R_A) || Y_{B_j}$ to device *B*.
7. Device *B* decrypts Y_{B_j} by its SUC_B to get $SUC^{-1}{}_B(Y_{Bj}) = X_{Bj}$.

8. Then device B decrypts S_A to get $F_{X_{Bj}}{}^{-1}(S_A) = R_A$.

9. Device B chooses a random R_B and computes the session key Z_{AB} as the hashed value:

$$Z_{AB} = H(R_A, R_B, X_{Bj}) \tag{8-8}$$

10. Device B sends R_B encrypted as S_B to device A by using X_{B_j} as a secret key:

$$S_B = F_{X_{B_j}}(R_B) \tag{8-9}$$

11. Device A can then decrypt S_B from (8-9) to get R_B and generate the same session key Z_{AB} as in (8-8).

Finally, the previous proposed protocol P_2 attains the following security functions:

- Device B with embedded SUC_B or user B is securely enrolled to the network though a trusted-device A.
- The devices A and B trust each other and share the same mutually authenticated session key Z_{AB}.
- The trusted link between A and B cannot be changed or faked later.
- This operation is required just once for each new edge.

8.3.3. P3: A Two-Step Device Authentication by a Mediators

Assume that A has a trusted link to the TA, devices A and B trust each other and have a trusted link, and the third device C is only linked to device B as depicted in Figure 8-12.

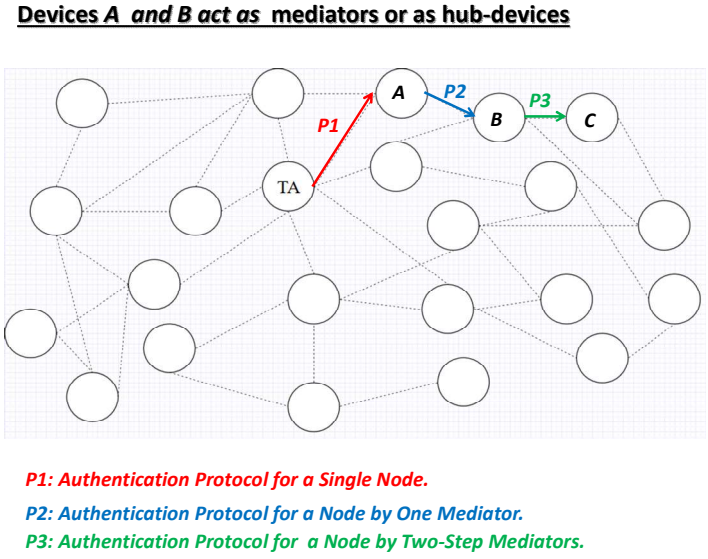


Figure 8-12. A Network with TA as a Central Hub and other possible Mediators such as A and B.

Here, the proposed protocol P_3 shows how device B can build a trusted link to device C and can authenticate C via the device A.

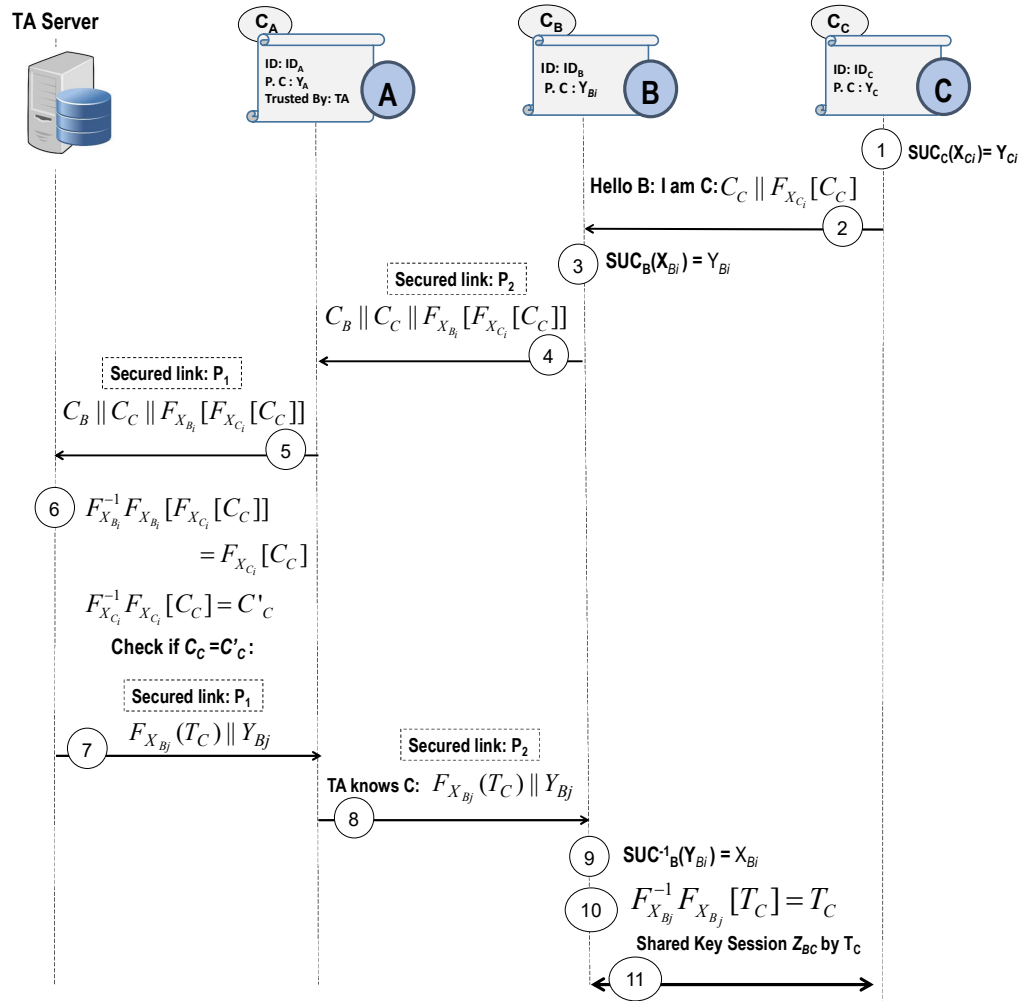


Figure 8-13. Authenticating a Device by Two-Steps Mediators. Adapted from [163].

Figure 8-13 shows the generated trusted link from *B* to *C*. The proposed protocol P_3 can be proceeded as follows:

1. Device *C* randomly chooses a cleartext entry X_{Ci} and uses its SUC_C to compute Y_{Ci}.
2. Device *C* computes and then sends $C_C \,||\, F_{X_{Ci}}[C_C]$ to device *B*.
3. A trusted-device *B* randomly chooses a value X_{Bi} and uses its SUC_B to compute Y_{Bi}.
4. A trusted-device *B* sends $C_B \,||\, C_C \,||\, F_{X_{Bi}}[F_{X_{Ci}}[C_C]]$ to device *A* asking to verify by the TA if the device *C* is authentic and trustable.
5. Trusted-device *A* forwards $C_B \,||\, C_C \,||\, F_{X_{Bi}}[F_{X_{Ci}}[C_C]]$ to the TA through a secured link asking to verify the device *C*.
6. The TA obtains the corresponding values X_{Ci}, X_{Bi} for Y_{Ci}, Y_{Bi} from *C*'s and *B*'s secret records, respectively. Then, the TA decrypts $F_{X_{Bi}}[F_{X_{Ci}}[C_C]]$ and verifies if $C'_C = C_C$.

7. TA selects a ticket T_C=(X_{Cj}, Y_{Cj}) from *C*'s records responding with an encrypted T_C exclusively for device *B* as $F_{XBj}(T_C) = F_{XBj}(X_{Cj}, Y_{Cj})$ and sends $F_{XBj}(T_C) \| Y_{Bj}$ back to device *B* via device *A*. Here, device *A* can't intercept the ticket.
8. Device *A* securely forwards $F_{XBj}(T_C) \| Y_{Bj}$ to device *B*.
9. Device *B* first uses its SUC_B to decrypt Y_{Bj} and gets X_{Bj}.
10. Then device *B* decrypts $F_{XBj}(T_C)$ to get T_C as $F^{-1}_{X_{Bj}} F_{X_{Bj}}(T_C) = T_C = (X_{Cj}, Y_{Cj})$.
11. Creating session key Z_{BC} between the device *B* and *C* proceeds correspondingly by using the previous protocol SSK.

Finally, the link between device *B* and device *C* uses Z_{BC} as a shared session key. Note that the proposed protocol P_3 can be generalized GP_3 to cover not only three connected devices but also *w* number of connected devices. If and only if *w*-1 devices have trusted links with the TA. This type of trusted chain was presented with other protocols in [163].

The proposed protocol P_3 attains the following security functions:

- Device *C* with embedded SUC_C or user *C* is securely enrolled to the network though a trusted- devices *A* and *B*.
- The trusted link between device *B* and device *C* is established and mutually authenticated.
- The trusted link between *B* and *C* cannot be changed or faked later.

Moreover, the trusted link involving this SUC technique is now ready to deploy in a network. In the following section, an algorithm deploying previous protocols is presented to build an authentication network [163].

8.4. Network Authentication Algorithm

In [171], the existing trust relationships between nodes/devices defines a so-called authentication graph. Following this work, an authenticated network is defined as all devices within a network being trusted by the TA.

Based on the previous protocols, a new algorithm (Algorithm.3) is developed, which deploys the protocols P_1 to P_3 to create authenticated networks as depicted in Figure 8-14. The resulting network is fully authenticated by the ownership of the devices incorporating the SUC. The links between the devices depend on the physical existence of the SUC.

Furthermore, the resulting authenticated network can also be considered as a network with several hubs. Such a network could contain a central hub such as TA together with *a few* connected devices to other devices that act as hubs. Moreover, the addition of new devices to the network can be carried out dynamically.

Algorithm.3 : Create an authenticated network [163]

Input: N nodes.

1: Connect m<<N nearest nodes to TA by using P_1.

2: Choose a node pair where one of m nodes and another belongs to the set of remainder nodes (N-m-1) and connect the chosen pair with a link by using P_2.

3: Reconnect the link to a randomly chosen node of (N-m-1) nodes by using P_3 and GP_3.

4: Repeat step 3 for all possible links in the network.

Figure 8-14 illustrates an authenticated network from a TA' point of view. Such a network includes a TA as a central hub surrounded by secondary and other hub nodes. Moreover, this network allows the TA to exchange authenticated data directly with trusted devices. For instance, if the TA would like to exchange data with an untrusted device, such as C, the TA could try to establish a new trust link with device C. This is a very costly procedure to establish a completely new link between the new device C and the TA. A practical solution would be, that the TA contacts a closely trusted device to device C, such as device B, and establishes a trusted link between device B and device C by using Z_{BC} as a shared session key.

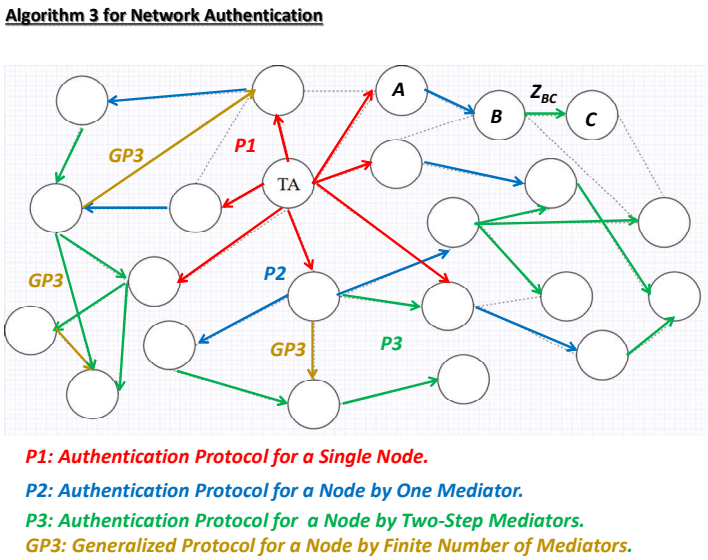


Figure 8-14. An Authenticated Network with TA as a Central Hub and other Possible Hub Nodes.

From a practical point of view, the properties of authenticated networks can provide a high level of security in designing a secure communicating system, especially for a large-scale network with hub nodes. In the following section, the security level of the resulting authenticated network using the SUC technique are analyzed and evaluated based on the recent attacks and threats

8.5. Security Analysis of the Proposed SUC-Protocols

SUC can be deployed in two different authentication schemes: First scheme, the authentication scheme deploying SUC-based key generation and the second scheme is SUC basic authentication protocol. For example, SUC is deployed as a key generation for device authentication in all previous proposed protocols. In particular, a unique secret unknown cipher SUC_A^{-1} is followed by $F_K(\cdot)$ as a keyed mapping using the secret key K extracted from SUC_A^{-1} as depicted in Figure 8-15. SUC-based basic authentication protocol was presented in the chapter 4 (section 4.1.3).

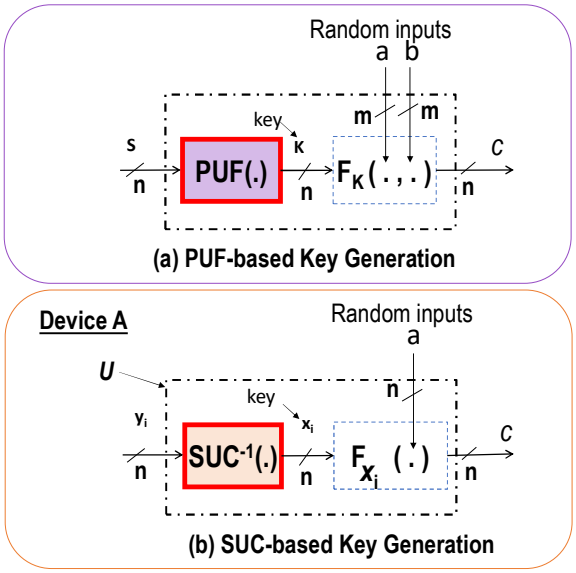


Figure 8-15. Model for Evaluating Identification Security. Adapted from [30].

In this section, the security analysis of authentication scheme deploying SUC-based key generation is investigated. The security analysis of such a scheme firstly requires defining the possible threats, and then applying the possible attacks on the proposed protocols. In the following section, the possible threats are determined, and the adversary model is defined.

8.5.1. Adversary Model

The target of an adversary is to take advantage of SUC-system drawbacks and to attack the system. For instance, an adversary tries to send remotely malicious instructions to impersonate a SUC-system. Furthermore, the adversary can be either an authorized device or an external adversary which applies attacks on the network. Therefore, there are two possible scenarios that can threaten the SUC-system. Firstly, cloning or faking a device with an embedded SUC. Secondly, the impersonation of a legitimate device.

In the following the adversary Ψ has (oracle) access to SUC-system [30]:

- Ψ knows the topology of the network.
- Ψ can run a device with embedded SUC.
- Ψ can know the design of the proposed protocols and can run them.

- Ψ knows the transmitted messages between two devices.
- Ψ can send a message to any device and the server S.
- Ψ can receive the device responses.
- Ψ can run a security experiment to try to impersonate any device.

8.5.2. Possible Attack Scenarios

Several attack scenarios can be applied on the proposed protocols such as modeling attacks, impersonation attack, etc.

- In the impersonation attack, the adversary collects the authentication data of a device and tries to assume/predict the device identity in a system or in a communication protocol.
- In the man-in-the-middle attack (MITM), an adversary collects the transmitted data between the devices in a system and uses them later to threaten the system. In SUC-system, the target of MITM adversary is to regenerate a valid shared session key created between two devices.
- In the modeling attacks, as mentioned in chapter 4, a response of the PUF for a given challenge can be predicted by the attacker [15]. This attack is equivalent to a collision attack on some known one-way hash function with a success probability less than $2^{-n/2}$ In the best case scenario, it is equivalent to a collision attack on some one-way function (possibly a weak one) with a success probability less than 2^{-n} [65]. Theoretically, the simulation of a response of the PUF requires at most $O(2^n)$ time/space resources complexity for a given challenge [15] [30].

Due to the modeling attacks being related to the SUC structure and plaintext/ciphertext pair behaviors, impersonation attack and MITM can be performed based on the designed protocols. Therefore, impersonation attacks and MITM are analyzed and evaluated in this section depending on the proposed adversary model.

A. Impersonation Attack

In [71], Sadeghi *et al.* showed that the successful impersonation attack on a PUF-based key generation (Figure 8-15) for device authentications is negligible in a RFID-system. Following this work, when a SUC plays the role of a key generator for a device authentication, as mentioned in all previous proposed protocols, the SUC generates a secret key of a standard cipher F.

Consider a device A with a unique secret unknown cipher SUC^{-1}_A. If this is followed by a standard cipher $F_K(.)$, as a keyed mapping using the secret key K extracted from SUC^{-1}_A, (see Figure 8-15). The structure of SUC^{-1}_A followed by F_K is theoretically used in all previous proposed protocols, where, the secret key K of F_K was derived from SUC as x_i.

Theorem. 8.1 [30]: The success probability of an impersonation attack on a device A with a unique secret unknown cipher SUC^{-1} is negligible for every adversary. Furthermore, a successful attack of one single device does not lead to all devices being cloned.

Proof:

Let $U: \{0,1\}^{2n} \rightarrow \{0,1\}^{n}$ be a cryptographic random mapping, where, U contains a cascade of $SUC^{-1}{}_{A}$ and F_K. Here, every SUC determines one mapping U. The security experiment with a parameter of U is described as follows:

For every adversary Ψ who interacts with a challenger C, where, C initializes l, n and $b \leftarrow \{0,1\}$, then, C also $b \leftarrow \{0,1\}$ initializes an oracle O^U that returns for input $p \in \{0,1\}^{2n}$ and an output $q \leftarrow U(p)$, if b=1, otherwise returns $q \leftarrow \{0,1\}^{n}$.

To run the "distinguishing experiment-1" (see chapter 1), Ψ generates a value $\tilde{a}$. Then, for any given y_i: $i \geq 0$, the adversary Ψ sends $p = (\tilde{a}, y_i)$ as an oracle query to C, for every $i \geq 0$. Here C responds either with $q = \tilde{c} = F_{x_i}(\tilde{a})$ or with $q \leftarrow \{0,1\}^{n}$.

Furthermore, the SUC_A as an invertible mapping has a probability of 2^{-n} guessing x_i from all 2^n values in the SUC input/output pairs space. This implies, that the SUC_A^{-1} acts as a key generator for the standard cipher F_K, $K=x_i$ and F_{xi} meets the requirements of PRF.

Thus, the advantage of Ψ to distinguish between $q = \tilde{c} \xleftarrow{U} F_{x_i}(\tilde{a})$ and $q = \tilde{c} \xleftarrow{U} \{0,1\}^{n}$ is defined as follows,

$$Adv_{PRF}^{f}(\Psi) = \left|\Pr[b = b'] - \tfrac{1}{2}\right| \tag{8-10}$$

Suppose that $\Pr[b = b']$ is not negligible. Then, there is at least an adversary Ψ who will always have an advantage in distinguishing between the output of U and a random value. In other words, Ψ can always determine $\tilde{c}$ such that $\tilde{c} = F_{x_i}(\tilde{a})$ for every $\tilde{a}$ and a given y_i. This implies that Ψ has an advantage in distinguishing between the output of PRF $F_{x_i}(.)$ and a random value, for every random key x_i derived by $SUC_A^{-1}(y_i)$: $i \geq 0$. This contradicts the definition 2.2 of the PRF $F_{x_i}(.)$.

As a result, the assumption that $\Pr[b = b']$ is not negligible is wrong. Therefore, $\Pr[b = b']$ can only be negligible. Therefore, the advantage of Ψ to distinguish between $q = \tilde{c} \xleftarrow{U} F_{x_i}(\tilde{a})$ and $q = \tilde{c} \xleftarrow{U} \{0,1\}^{n}$ is negligible.

Suppose that if the value $U(\tilde{a}, y_i)$ can be determined for some specific $i=i_0$, then,

$$\Pr[b = b'] = \frac{(2^n)^{\sigma \cdot 2^n - 1}}{(2^n)^{\sigma \cdot 2^n}} = \frac{1}{2^n} \tag{8-11}$$

where, σ is the number of SUCs. Therefore, if the adversary successfully attacks one SUC, then this will not lead to-all SUCs being attacked.

It turns out that the adversary cannot impersonate (simulate) a SUC based on the input/output pairs behavior. Therefore, the SUC provides a system with a security bound of $O(2^n)$.

B. Man-in-the-middle Attack

In this attack scenario, the adversary can interpret the transmitted messages between parties in a communication protocol. Here the adversary has independent connections with the parties, where, the target of MITM adversary is to regenerate a valid shared session key created between two parties. In the following MITM is applied on the proposed protocol P_1 for a single device. Other designed protocols can be analyzed in a similar way.

The MITM adversary to the proposed protocol P_1 tries to regenerate $Z_{A\text{-}TA}$ in (8-6) between TA and device A. Therefore, the successful MITM attack is equivalent to a successful prediction of (R_A, R_{TA}, X_{Aj}) that allows the adversary to generate $Z_{A-TA} = H(R_A, R_{TA}, X_{Aj})$. In this case, it is enough to guess/predict X_{Aj} in steps 3, 5, or 7 of the SSK protocol. Meanwhile, the SUC is defined as an invertible cryptographic function with a significant level of security. Therefore, the probability of predicting of one SUC input is 2^{-n}, where, n is the SUC input size. This implies that the successful MITM complexity is $O(2^n)$.

In conclusion, MITM attacks do not work on the proposed protocols, because, breaking the proposed protocols is equivalent to breaking the SUC.

8.6. Summary

In this chapter, SUCs, as physical security anchors, are deployed to propagate the trust chain in communication networks. For the first time the proposed generic protocols illustrate how a trust chain can be established and extended in a network with single or multiple hubs. On the other hand, the proposed generic protocols are considered the first step towards a authenticated network, where, the successful attack on hubs cannot be passed to other devices in such a network.

Furthermore, SUC-based key generation offers higher security level compared to the PUF-based key generation. The theoretical security bounds attained when SUC-based key generation is deployed for key generation protocols. it has been proved that SUC-based key generation attains higher security bounds compared with the conventional PUF-based key-generation [30] The SUC-based system reduces dramatically the required memory storage of CRPs.

"*We ought not to be embarrassed of appreciating the truth and of obtaining it wherever it comes from.... Nothing should be dearer to the seeker of truth than the truth itself.*"

On the Definitions of Things and their Description

Al-Kindi (801–873)

9. CONCLUSION AND FUTURE WORK

Throughout this thesis a new approach to create a digital clone-resistant physical identity was introduced. This so-called SUC can be perceived as a DNA-like identity for electronic devices. SUC is proposed to serve as a resilient digital PUF in future smart electronic devices. This thesis shows that SUC as a pure digital mapping presents a new alternative to the PUF-technology.

Currently, one of the possible practical approaches to implement SUCs is to mutate them in self-reconfigurable non-volatile SoC FPGA devices. Here, a self-reconfigurable SoC architecture is a basic required technology for mutating secret ciphers and non-volatile memory is required to permanently store the hardwired secret ciphers. The results showed that selecting SUC from a class of ciphers with very high cardinality makes cloning attacks almost impossible. Additionally, the SUC-design as a PRF prohibits modeling attacks. Here, a self-generated SUC inside a chip is modeled as a secure PRP randomly chosen from a huge class of ciphers.

In order to realize SUC new hardware-oriented classes of ciphers were designed as SUC-possible structures. The major building blocks in the design strategy are FPGA hard-core multipliers and LUTs. By deploying such multipliers, near to zero-cost can be attained if the SUC module is embedded in a FPGA device which have unused hardware resources.

In this thesis new secret unknown ciphers or SUCs were developed, in particular:

- The SUC concept and its usage were first introduced.
- New hardware-oriented classes of ciphers were designed by deploying multipliers/Mathblocks as basic building elements.
- Security analysis of the SUCs-designed classes was evaluated.
- Security applications deploying SUCs as a security anchor in contemporary systems were investigated.

In chapter 3, PUFs were investigated as examples of useful unknown functions. PUFs are defined as fully unknown and usable hash-functions. On the other hand, PUF-based key generation and a block cipher based on PUF were discussed. The description and classification of security threats on these proposals were presented for later comparison. Finally, PUF-authentication protocols were discussed and analyzed.

In chapter 4, the SUC concept was introduced. This concept is considered to be a Digital-PUF implemented in a modern FPGA. Thus, the targeted implementation environment and its requirements were proposed and discussed. Therefore, the most challenging part in the SUC concept is devising an efficient and low-cost up to date realization technique. This chapter covered a simple model of SUCs, and the results show that SUC offers a new solution overcoming the drawbacks of PUFs, since, it is non-reversible and very hard to predict.

In chapter 5, a possible SUC implementation strategy was presented, where, converting smart programmable VLSI devices into physically hard-to-clone devices was devised. This strategy allowed to generate new special huge classes of FPGA-optimized ciphers. The targeted VLSI technologies were proposed and investigated such as the smart self-reconfiguring and non-volatile FPGA devices. The suggested strategy is conducted based on particularly the use of non-consumed Mathblock resources in programmable SoC devices.

Next, in chapters 6 and 7, two different designs of SUCs were introduced. In the first proposal, a Feistel-like cipher was presented with two different proposals of implementation. The second proposal included a new cipher design as a cascade of ciphers. Such a design benefits from the proposed implementation strategy to convert future programmable VLSI devices into physically clone-resistant devices. Hardware and software complexities for realizing such proposals were optimized and evaluated for a sample expected target technology. The attained security levels of the resulting SUCs were evaluated and shown to be scalable and secure against post-quantum crypto systems.

In chapter 8, several generic authentication protocols were discussed. The key result was to show how to build trust relation between finite number of nodes/ devices when one of them acts as a mediator node. Such technique did not have an influence on the proposed chain of nodes/devices. The main target of this chapter was to present the practical and efficient trust-chaining in large networks when the physically clone-resistant identity is involved.

9.1. Possible Applications Outside the Scope of this Thesis

This thesis is a part of an ongoing basic research towards smart SUCs and their future applications. Being created in a manufacturer-independent-process and by end-users, the technology is expected to be attractive for wide spectrum of applications in future automotive and IoT environments. In fact, there is a lot of applications considering the contribution of this thesis. Several applications deploying SUCs have been published recently. Such applications are shown in the following examples:

- **SUC for Securing Smartphones** [174]: The proposed approach combines two keys: Firstly, SUC converting a Smartphone to a clone-resistant device. Secondly, a unique biometric key extracted from a smartphone user during moving his/her hand up and down. Particularly, embedding a SUC in the smartphone device prohibits the cloning of the smartphone and the detection of the biometric key generates a unique user identity. The resulting joint user-smartphone clone-resistant identity ensures a high level of security for the communication between a user/smartphone and the server.
- **SUC for Secured Fleet Management System** [175]: A secured fleet management system can be attained by embedding SUCs in the fleet components/entities. This implies that several joint-physical identification protocols can be designed. Such protocols deploying SUCs prevent illegal replacement of Vehicles and transported goods, where, vehicles and goods are uniquely and securely identifiable by their SUCs. The security level of this technique can be evaluated by applying several attacks on modern networked fleet management systems.
- **SUC for Non-Repeatable Clone-Resistant Group of Devices** [176]: A very valuable scenario for cloud/fog networks can be presented by deploying SUCs. A new non-repeatable group of devices can be constructed, where, SUCs ensure secured communicating between the group of devices with solid unclonable-ciphering in the sense that no physical unit within the group can be replaced by anybody. In this created group, each device has a common unclonable identity with other devices in the group in addition to their own individual unclonable identity. Here, there is no latter possibility to increase the number of devices and their identities.

9.2. Ongoing and Future Work

In this thesis, the SUC is proposed to provide electronic devices with unique clone-resistant digital signatures or a clone-resistant identity. A secure identity is considered as the backbone of future security system to protect data transmission. The common identity usages serve to check and verify the integrity of devices, to control the processes and access to the devices, and to meet traceability requirements.

On the other hand, facilitating the SUC deployment in security systems requires more

investigation and study. Therefore, the following essential future challenges should be taken into consideration:

- Developing a practical approach of the proposed GENIE.
- Designing new hardware-oriented classes of ciphers to server as SUCs.
- Devising other alternative implementations of the SUC proposed classes in future reconfigurable VLSI devices.
- Deploying SUCs in different security applications and platforms.
- The need for applying several attacks on the proposed SUCs to detect the cipher-design drawbacks is recognized.

Within our proposed SUC designs, we did not apply physical attack scenarios and we did not explain how to apply side channel attacks on a SUC in the targeted environment. The cost of such attacks provides us with more reliable indicators of the SUC-security levels. Therefore, future directions of this research can be envisaged to cover physical attacks, that lead later to developing a good implementation strategy of the designed SUCs. Here, continuous improvements of the proposed ciphers classes are required, as SUC design is a multidisciplinary and very challenging task

Finally, although we have presented only some application protocols for SUC-based identity, SUC can be used a security anchor for a large spectrum of applications. To achieve that, the possible security applications deploying SUCs should be more investigated and envisaged. Furthermore, studying the usage of SUCs for IoT applications needs to be researched. For instance, vehicular networks are very important fields to deploy SUCs.

To conclude, our published works and papers show that SUC is indeed considered a relevant and efficient solution to several security problems in emerging technologies. In this direction, designing SUCs and SUC-systems is very hard to be covered in one work. Therefore, the need for complementary works is very necessary.

APPENDIX A: PUBLICATIONS

The author submitted 14 papers. 11 of them related directly to this thesis and the rest as *applications of the proposed SUC*:

This appendix lists the author publications that are involved in this thesis.

1. **Mulhem, S.,** Mars, A., and W. Adi, "**Low-Complexity Nonlinear Self-Inverse Permutation for Physically Clone-Resistant Identities**," Cryptography,2020.
2. **Mulhem, S.** and W. Adi, "**New Mathblocks-Based Feistel-Like Ciphers for Creating Clone-Resistant FPGA Devices**," Cryptography, 2019.
3. **Mulhem, S**, Ayache, M, and Adi, W, "**Mini-Block-Baesd Cipher Class for Physical Clone- Resistant Devices**", in eighth IEEE International Conference on Emerging Security Technologies - EST July 2019, At Cochester, Essex, UK.
4. **Mulhem, S**, Mohammad, M and Adi, W, "**A New Low-Complexity Cipher Class for Clone-Resistant Identities**", in CTS - Computers in Technical Systems, 42nd International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), May, 2019.
5. Zarrouk, R, **Mulhem, S**, Reich,L , and Adi, W, "Non-Repeatable Clone-Resistant Group Device-Identity", International Conference on Cyber Security for Emerging Technologies (CSET'19), Oct, 2019, Doha, Qatar.

6. **Mulhem, S,** Mars,A, and Adi,W, "**A Cipher Class Based on Golden S-Boxes for Creating Clone-Resistant Identities**" in IOSes 2018 International workshop on Information & Operational Technology (IT & OT) security systems, Sep, 2018.
7. **Mulhem, S,** Zarrouk,R, and Adi,W, "**Security and Complexity Bounds of SUC-Based Physical Identity**" in 12th NASA/ESA Conference on Adaptive Hardware and Systems, Aug, 2018.
8. **Mulhem,S**, Ahmad,A, and Adi,W," Accelerometer–Based Joint User-Device Clone-Resistant Identity" ,in World Conference on Smart Trends in Systems, Security and Sustainability (WS4 2018), Oct, 2018.
9. Hamadaqa, E, **Mulhem, S,** Mars, A, and Adi, A, "Clone-Resistant Joint Identity Technique for Secured Fleet Management Systems," in 12th NASA/ESA Conference on Adaptive Hardware and Systems, 2018.
10. **Mulhem, S,** Adi,W, Mars,A, and Prevelakis,V,"**Chaining Trusted Links by Deploying Secured Physical Identities**", Conference: Seventh IEEE International Conference on Emerging Security Technologies - EST September 2017, At Canterbury, UK.
11. Adi, W, **Mulhem, S** and Mars, A, "**Secured Remote Sensing by Deploying Clone-Resistant Secret Unknown Ciphers**", Conference: 2017 IEEE International Conference on Consumer Electronics, Volume: Special Session 25_Cyber Security and Privacy of Mobile, Web and Internet of Things (SS-CWT), June 2017.

This appendix lists the author publications as applications that are related to this thesis but not involved in:

1. Mars, A, Adi, W, **Mulhem, S** and Hamadaqa, E,"**Random Stream Cipher as a PUF-like Identity in FPGA Environment**", Conference: Seventh IEEE International Conference on Emerging Security Technologies - EST September 2017, At Canterbury, UK.
2. Hamadaqa, E, Mars,Adi, W, and **Mulhem, S,** "**A Clone-Resistant Vehicular RKE by Deploying SUC**", Conference: Seventh IEEE International Conference on Emerging Security Technologies - EST September 2017, At Canterbury, UK.
3. Adi, W, Mars, A and **Mulhem, S**, "**Generic Identification Protocols by Deploying Secret Unknown Ciphers (SUCs)**", conference: 2017 IEEE International Conference on Consumer Electronics-(ICCE-TW), At Taiwan, Volume: Special Session 07_Advanced Cryptography and Its Applications,june 2017.

APPENDIX B:

In the following, we prove that using Maurer's simplified treatment suggested in [114], the proposed Feistel-like cipher $\zeta(g, f, f)$ is a pseudorandom permutation. The same proof was proposed in [29].

Assume that the proposed Feistel-like cipher is $\zeta(g, f, f)$, where, $g, f \xleftarrow{U} F_n$,then $\zeta(g, f, f)$ (L_i, R_i) is described as :

$$\left.\begin{aligned} X_i &= aL_i + bg(R_i) \\ S_i &= aR_i + bf(X_i) \\ T_i &= aX_i + bf(S_i) \end{aligned}\right\} \quad \text{(B-1)}$$

To prove $\zeta(g, f, f)$ is a pseudorandom permutation. Distinguish Experiment-2 should be performed for $\zeta(g, f, f)$. Such experiment is hence an immediate consequence of the following Lemma.

Lemma B.1: For every function and for any q pairs (L_i, R_i) from $\{0,1\}^n \times \{0,1\}^n$, $i=1,\ldots q$.

$$\left|\Pr\left[G\left(\zeta(g,f,f)(L_1,R_1),\cdots,\zeta(g,f,f)(L_k,R_k)\right)=1 : f \xleftarrow{U} F_n\right] - \Pr_G\right| \le \frac{q^2}{2^n} \quad \text{(B-2)}$$

Where, $g, f \xleftarrow{U} F$ and $\Pr_G$ defined as:

$$\Pr_G = \frac{\#\{(x_1,\cdots,x_k) \in \left(\{0,1\}^{2n}\right)^q : G(x_1,\cdots,x_q) = 1\}}{2^{2nq}} \tag{0-1}$$

Proof:

Assume without loss of generality that the q pairs (L_i, R_i) are distinct. According to (B-1), the outputs of the first, second, and third round are (R_i, X_i), (X_i, S_i), and (S_i, T_i), respectively. Let A_X be the event that $\{X_i\}_{i=1}^q$ are distinct and let A_S be the event that $\{S_i\}_{i=1}^q$ are distinct. Then, $A_X \cap A_S$ is the event that $\{X_i\}_{i=1}^q$ and $\{S_i\}_{i=1}^q$ are distinct.

Now, if the event A_X occurs, then the values $S_i = aR_i + bf(X_i)$ are random for $i=1,\ldots q$, where $f \xleftarrow{U} F_n$. On the other hand, $g \xleftarrow{U} F_n$ and if the event A_S occurs, then the values $T_i = aX_i + bg(S_i)$ are random for $i=1,\ldots q$. In this case, $\zeta(g,f,f)$ behaves precisely like a randomly chosen function from the set of all possible functions F_{2n} from $\{0,1\}^{2n}$ to $\{0,1\}^{2n}$, and the probability of distinguishing between $\zeta(g,f,f)$ and a random function from F_{2n} is:

$$\left|\Pr\left[G\left(\zeta(g,f,f)(L_1,R_1),\cdots,\zeta(g,f,f)(L_k,R_k)\right) = 1 : f \xleftarrow{U} F_n\right] - \Pr_G\right| \le 1 - \Pr[A_X \cap A_S] \tag{B-3}$$

And,

$$1 - \Pr[A_X \cap A_S] = \Pr[\overline{A_X \cap A_S}] = \Pr[\overline{A_X} \cup \overline{A_S}] \le \Pr[\overline{A_X}] + \Pr[\overline{A_S}] \tag{B-4}$$

Where, $\overline{A_X}$ ($\overline{A_S}$) in the complementary event of A_X (A_X) occurring when $\{X_i\}_{i=1}^q$ ($\{S_i\}_{i=1}^q$) are not distinct, respectively.

For $i \neq j$, and according to the main assumption: the q pairs (L_i, R_i) are distinct.

$$\Pr[\overline{A_X}] = \binom{q}{2} \sum_{1 \le i < j \le q} \Pr[X_i = X_j], \text{and } \Pr[\overline{A_S}] = \binom{q}{2} \sum_{1 \le i < j \le q} \Pr[S_i = S_j] \tag{B-5}$$

Where, $\binom{q}{2}$ is the number of choosing 2 equal values $[X_i = X_j]$ ($[S_i = S_j]$) out of q from $\overline{A_X}$ ($\overline{A_S}$), respectively. On the other hand, the q pairs (L_i, R_i) are distinct by assumption, $\Pr[X_i = X_j]$ and $\Pr[S_i = S_j]$ are computed as,

$$\Pr[X_i = X_j] = \begin{cases} 2^{-n}; & R_i \neq R_j \\ 0 \quad ; & R_i = R_j \end{cases}, \text{and} \quad \Pr[S_i = S_j] = \begin{cases} 2^{-n}; & X_i \neq X_j \\ 0 \quad ; & X_i = X_j \end{cases} \tag{B-6}$$

From (B-6) and (B-5),

$$\Pr[\overline{A_X}] = \binom{q}{2} \cdot 2^{-n}, \text{and } \Pr[\overline{A_S}] = \binom{q}{2} \cdot 2^{-n} \tag{B-7}$$

Substituting (B-7) by (B-4),

$$1-\Pr[A_X \cap A_S] \le 2\cdot\binom{q}{2}\cdot 2^{-n} = \frac{q(q-1)}{2^n} < \frac{q^2}{2^n} \quad \text{(B-8)}$$

Call (B-3), we obtain,

$$\left|\Pr\left[G\left(\zeta(g,f,f)(L_1,R_1),\cdots,\zeta(g,f,f)(L_k,R_k)\right)=1 : f \xleftarrow{U} F_n\right]-\Pr_G\right| \le \frac{q^2}{2^n}$$

□

Distinguish Experiment-2 for $\zeta(g, f, f)$:

Step 1: For the proposed Feistel-like cipher $\zeta(g,f,f)$ defined over (K, X), where $|X| = 2^{2n}$. Consider an adversary (distinguisher) Ψ that interacts with a challenger C who works as follows:

- C randomly chooses one bit $b \xleftarrow{U} \{0,1\}$.
- C returns $P \xleftarrow{U} B_{2n}$ if b=1 to Ψ, otherwise returns $P \xleftarrow{D} \zeta(g,f,f)$, where $g, f \xleftarrow{U} F_n$.

Within time t.

Step 2: The adversary Ψ submits to challenger C a polynomial number of queries (q) such as (L_i, R_i) from $\{0,1\}^n \times \{0,1\}^n$, i=1,…q.

Step 3: The adversary terminates the experiment by returning b'.

According to lemma B.1, the advantage of Ψ to distinguish between $\zeta(g, f, f)$ and a random function is:

$$Adv_{PRF}^{\zeta(g,f,f)}(\Psi) \le \frac{q^2}{2^n} \quad \text{(B-9)}$$

Now, the PRF Switching Lemma (lemma 2.1) stated that,

$$Adv_{PRP}^{\zeta(g,f,f)}(\Psi) < Adv_{PRF}^{\zeta(g,f,f)}(\Psi) + q^2 / 2^{n+1} \quad \text{(B-10)}$$

Observe that if q is not of order 2^n, it is not possible to distinguish $\zeta(g, f, f)$ from a random permutation with a high probability.

Lemma 4.5: For $q<2^n$, the advantage of Ψ to distinguish between a random permutation $P \xleftarrow{U} B_{2n}$ and the proposed cipher $P \xleftarrow{D} \zeta(g,f,f)$, where $g, f \xleftarrow{U} F$, is

$$Adv_{PRP}^{\zeta(g,f,f)}(\Psi) < 3q^2 / 2^{n+1} \quad \text{(B-11)}$$

Furthermore, the previous lemma concludes the main result by a similar argument.

BIBLIOGRAPHY

[1] W. Adi and B. Soudan, "Bio-Inspired Electronic-Mutation with genetic properties for Secured Identification," in *ECSIS Symposium on Bio-inspired, Learning, and Intelligent Systems for Security*, 2007, pp. 133–136.

[2] R. Maes and I. Verbauwhede, "Physically Unclonable Functions: A Study on the State of the Art and Future Research Directions," Springer, Berlin, Heidelberg, 2010, pp. 3–37.

[3] F. Galton, *Finger Prints*. Macmillan, 1892.

[4] B. William J. Herschel, *The Origin of Finger-Printing*. London: Oxford University Press, 1916.

[5] Tolk and K. M, "Reflective particle technology for identification of critical components," Orlando, FL (United States), Jan. 1992.

[6] S. Mittal and J. S. Vetter, "A Survey of Software Techniques for Using Non-Volatile Memories for Storage and Main Memory Systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 5, pp. 1537–1550, May 2016.

[7] D. A. Patterson and J. L. Hennessy, *Computer organization and design : the hardware/software interface.*, 5th Edition. USA: Elsevier , 2013.

[8] S. Skorobogatov, "Semi-invasive attacks: a new approach to hardware security analysis," University of Cambridge, 2005.

[9] T. Idriss, H. Idriss, and M. Bayoumi, "A PUF-based paradigm for IoT security," in *2016 IEEE 3rd World Forum on Internet of Things (WF-IoT)*, 2016, pp. 700–705.

[10] J. Delvaux, R. Peeters, D. Gu, and I. Verbauwhede, "A Survey on Lightweight Entity Authentication with Strong PUFs," *ACM Comput. Surv.*, vol. 48, no. 2, pp. 1–42, Oct. 2015.

[11] Y. Dodis, L. Reyzin, and A. Smith, "Fuzzy Extractors: How to Generate Strong Keys from Biometrics and Other Noisy Data," in *Advances in Cryptology - EUROCRYPT 2004*, 2004, pp. 523–540.

[12] W. Adi and B. Soudan, "Electronic mutation technology and secured identification," in *2007 9th International Symposium on Signal Processing and Its Applications*, 2007, pp. 1–4.

[13] W. Adi, "Clone-Resistant DNA-Like Secured Dynamic Identity," in *2008 Bio-inspired, Learning and Intelligent Systems for Security*, 2008, pp. 148–153.

[14] C. Helfmeier, C. Boit, D. Nedospasov, and J.-P. Seifert, "Cloning Physically Unclonable Functions," in *2013 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*, 2013, pp. 1–6.

[15] U. R ührmair, F. Sehnke, J. S ölter, G. Dror, S. Devadas, and J. ürgen Schmidhuber, "Modeling attacks on physical unclonable functions," in *Proceedings of the 17th ACM conference on Computer and communications security - CCS '10*, 2010, p. 237.

[16] W. Adi, "Autonomous Physical Secret Functions and Clone-Resistant Identification," in *2009 Symposium on Bio-inspired Learning and Intelligent Systems for Security*, 2009, pp. 83–88.

[17] A. Bogdanov, L. R. Knudsen, G. Leander, F.-X. Standaert, J. Steinberger, and E. Tischhauser, "Key-Alternating Ciphers in a Provable Setting: Encryption Using a Small Number of Public Permutations," Springer, Berlin, Heidelberg, 2012, pp. 45–62.

[18] H. Feistel, "Cryptography and Computer Privacy," *Sci. Am.*, vol. 228, pp. 15–23, 1973.

[19] H. Feistel, W. A. Notz, and J. L. Smith, “Some cryptographic techniques for machine-to-machine data communications,” *Proc. IEEE*, vol. 63, no. 11, pp. 1545–1554, 1975.

[20] J. Daemen and V. Rijmen, *The Design of Rijndael : AES - The Advanced Encryption Standard*. Springer Berlin Heidelberg, 2002.

[21] National Bureau of Standards, “FIPS publication 46: Data Encryption Standard (DES),” Jan. 1977.

[22] C. E. Shannon, “Communication Theory of Secrecy Systems*,” *Bell Syst. Tech. J.*, vol. 28, no. 4, pp. 656–715, Oct. 1949.

[23] O. Goldreich, S. Goldwasser, and S. Micali, “How to construct random functions,” *J. ACM*, vol. 33, no. 4, pp. 792–807, Aug. 1986.

[24] O. Goldreich, S. Goldwasser, and S. Micali, “On the Cryptographic Applications of Random Functions (Extended Abstract),” in *Advances in Cryptology*, Berlin, Heidelberg: Springer Berlin Heidelberg, 1984, pp. 276–288.

[25] D. Boneh and V. Shoup, *A Graduate Course in Applied Cryptography*, 0.4. Stanford, California, USA: Stanford University, 2017.

[26] A. Bogdanov and A. Rosen, “Pseudorandom Functions: Three Decades Later,” in *Tutorials on the Foundations of Cryptography. Information Security and Cryptography.*, Lindell Y., Ed. Springer, Cham, 2017, pp. 79–158.

[27] L. G. Valiant and L. G., “A theory of the learnable,” *Commun. ACM*, vol. 27, no. 11, pp. 1134–1142, Nov. 1984.

[28] R. L. Rivest, “Cryptography and machine learning,” Springer, Berlin, Heidelberg, 1993, pp. 427–439.

[29] S. Mulhem and W. Adi, “New Mathblocks-Based Feistel-Like Ciphers for Creating Clone-Resistant FPGA Devices,” *Cryptography*, no. Hardware Security, 2019.

[30] S. Mulhem, R. Zarrouk, and W. Adi, “Security and Complexity Bounds of SUC-Based Physical Identity,” in *2018 NASA/ESA Conference on Adaptive Hardware and Systems (AHS)*, 2018, pp. 317–322.

[31] S. Mulhem, A. Mars, and W. Adi, “Low-Complexity Nonlinear Self-Inverse Permutation for Physically Clone-Resistant Identities,” *Cryptogr.* , 2019.

[32] R. Pappu, B. Recht, J. Taylor, and N. Gershenfeld, “Physical One-Way Functions,” *Science (80-.).*, vol. 297, no. 5589, pp. 2026–2030, Sep. 2002.

[33] P. Tuyls and L. Batina, “RFID-Tags for Anti-counterfeiting,” Springer, Berlin, Heidelberg, 2006, pp. 115–131.

[34] Daihyun Lim, J. W. Lee, B. Gassend, G. E. Suh, M. van Dijk, and S. Devadas, “Extracting secret keys from integrated circuits,” *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 13, no. 10, pp. 1200–1205, Oct. 2005.

[35] J. Guajardo, S. S. Kumar, G.-J. Schrijen, and P. Tuyls, “FPGA Intrinsic PUFs and Their Use for IP Protection,” in *Cryptographic Hardware and Embedded Systems - CHES 2007*, Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 63–80.

[36] F. Armknecht, R. Maes, A.-R. Sadeghi, B. Sunar, and P. Tuyls, “Memory Leakage-Resilient Encryption Based on Physically Unclonable Functions,” in *Advances in Cryptology – ASIACRYPT 2009. ASIACRYPT 2009*, 2009, pp. 685–702.

[37] T. Xu and M. Potkonjak, “Robust and flexible FPGA-based digital PUF,” in *2014 24th International Conference on Field Programmable Logic and Applications (FPL)*, 2014, pp. 1–6.

[38] J. Wu and M. O'Neill, "On Foundation and Construction of Physical Unclonable Functions," *IACR Cryptol. ePrint Arch.*, p. 171, 2010.

[39] G. E. Suh and S. Devadas, "Physical unclonable functions for device authentication and secret key generation," in *Proceedings of the 44th annual conference on Design automation - DAC '07*, 2007, p. 9.

[40] B. Gassend, D. Clarke, M. van Dijk, and S. Devadas, "Controlled physical random functions," in *18th Annual Computer Security Applications Conference, 2002. Proceedings.*, pp. 149–160.

[41] B. L. P. Gassend, "Physical random functions," Massachusetts Institute of Technology, 2003.

[42] C. Marchand, L. Bossuet, U. Mureddu, N. Bochard, A. Cherkaoui, and V. Fischer, "Implementation and Characterization of a Physical Unclonable Function for IoT: A Case Study With the TERO-PUF," *IEEE Trans. Comput. Des. Integr. Circuits Syst.*, vol. 37, no. 1, pp. 97–109, Jan. 2018.

[43] J. Guajardo, S. S. Kumar, G.-J. Schrijen, and P. Tuyls, "FPGA Intrinsic PUFs and Their Use for IP Protection," in *Cryptographic Hardware and Embedded Systems - CHES 2007*, Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 63–80.

[44] U. Rührmair, J. Sölter, and F. Sehnke, "On the Foundations of Physical Unclonable Functions," *IACR Cryptol. ePrint Arch.*, 2009.

[45] D. E. Holcomb, W. P. Burleson, and K. Fu, "Power-Up SRAM State as an Identifying Fingerprint and Source of True Random Numbers," *IEEE Trans. Comput.*, vol. 58, no. 9, pp. 1198–1210, Sep. 2009.

[46] C. Böhm and M. Hofer, *Physical Unclonable Functions in Theory and Practice.* Springer, 2012.

[47] M. Barbareschi, P. Bagnasco, and A. Mazzeo, "Authenticating IoT Devices with Physically Unclonable Functions Models," in *2015 10th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC)*, 2015, pp. 563–567.

[48] C. Bösch, J. Guajardo, A.-R. Sadeghi, J. Shokrollahi, and P. Tuyls, "Efficient Helper Data Key Extractor on FPGAs," in *Cryptographic Hardware and Embedded Systems – CHES 2008*, Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 181–197.

[49] J. Delvaux, D. Gu, D. Schellekens, and I. Verbauwhede, "Helper Data Algorithms for PUF-Based Key Generation: Overview and Analysis," *IEEE Trans. Comput. Des. Integr. Circuits Syst.*, vol. 34, no. 6, pp. 889–902, Jun. 2015.

[50] Y. Dodis, R. Ostrovsky, L. Reyzin, and A. Smith, "Fuzzy Extractors: How to Generate Strong Keys from Biometrics and Other Noisy Data," *SIAM J. Comput.*, vol. 38, no. 1, pp. 97–139, Jan. 2008.

[51] Y. Wang, X. Xi, and M. Orshansky, "Lattice PUF: A Strong Physical Unclonable Function Provably Secure against Machine Learning Attacks," Sep. 2019.

[52] A. Babaei and G. Schiele, "Physical Unclonable Functions in the Internet of Things: State of the Art and Open Challenges," *Sensors*, vol. 19, no. 14, p. 3208, Jul. 2019.

[53] U. Rührmair, H. Busch, and S. Katzenbeisser, "Strong PUFs: Models, Constructions, and Security Proofs," in *Towards Hardware-Intrinsic Security. Information Security and Cryptography.*, N. D. (Sadeghi AR., Ed. Springer, Berlin, Heidelberg, 2010, pp. 79–96.

[54] U. Ruhrmair and M. van Dijk, "PUFs in Security Protocols: Attack Models and Security Evaluations," in *2013 IEEE Symposium on Security and Privacy*, 2013, pp. 286–300.

[55] B. Gassend, D. Clarke, M. van Dijk, and S. Devadas, "Silicon physical random functions,"

in *Proceedings of the 9th ACM conference on Computer and communications security - CCS '02*, 2002, p. 148.

[56] U. Rührmair, C. Jaeger, M. Bator, M. Stutzmann, P. Lugli, and G. Csaba, "Applications of High-Capacity Crossbar Memories in Cryptography," *IEEE Trans. Nanotechnol.*, vol. 10, no. 3, pp. 489–498, May 2011.

[57] G. Becker, "Intentional and unintentional side-channels in embedded systems," University of Massachusetts Amherst, United States., 2014.

[58] D. Merli, D. Schuster, F. Stumpf, and G. Sigl, "Side-Channel Analysis of PUFs and Fuzzy Extractors," Springer, Berlin, Heidelberg, 2011, pp. 33–47.

[59] G. T. Becker and R. Kumar, "Active and Passive Side-Channel Attacks on Delay Based PUF Designs," *undefined*, 2014.

[60] A. Mahmoud, U. Rührmair, M. Majzoobi, and F. Koushanfar, "Combined Modeling and Side Channel Attacks on Strong PUFs," 2013.

[61] L. Santiago *et al.*, "Realizing strong PUF from weak PUF via neural computing," in *2017 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*, 2017, pp. 1–6.

[62] L. Santiago de Araújo *et al.*, "Design of Robust, High-Entropy Strong PUFs via Weightless Neural Network," *J. Hardw. Syst. Secur.*, vol. 3, no. 3, pp. 235–249, Sep. 2019.

[63] S. Hou, Y. Guo, and S. Li, "A Lightweight LFSR-Based Strong Physical Unclonable Function Design on FPGA," *IEEE Access*, vol. 7, pp. 64778–64787, 2019.

[64] J. Massey, "Shift-register synthesis and BCH decoding," *IEEE Trans. Inf. Theory*, vol. 15, no. 1, pp. 122–127, Jan. 1969.

[65] O. Goldreich, *Foundations of cryptography*. Cambridge University Press, 2003.

[66] J. D. Bekenstein, "How does the Entropy/Information Bound Work?," *Found. Phys.*, vol. 35, no. 11, pp. 1805–1823, Nov. 2005.

[67] L. P. Hyvärinen, *Information Theory for Systems Engineers*. Springer Berlin Heidelberg, 1968.

[68] A. Schaub, O. Rioul, Joseph, and J. J. Boutros, "Entropy Estimation of Physically Unclonable Functions via Chow Parameters," *Comput. Sci. Inf. Theory*, Jul. 2019.

[69] S. Katzenbeisser, Ü. Kocabaş, V. Rožić, A.-R. Sadeghi, I. Verbauwhede, and C. Wachsmann, "PUFs: Myth, Fact or Busted? A Security Evaluation of Physically Unclonable Functions (PUFs) Cast in Silicon," in *Cryptographic Hardware and Embedded Systems – CHES 2012. CHES 2012. Lecture Notes in Computer Science, vol 7428.*, 2012, pp. 283–301.

[70] M. Bhargava and K. Mai, "An efficient reliable PUF-based cryptographic key generator in 65nm CMOS," in *2014 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2014, pp. 1–6.

[71] A. Sadeghi, I. Visconti, and C. Wachsmann, "PUF-enhanced RFID security and privacy," in *In 2nd Work-shop on Se-cu-re Com-po-nent and Sys-tem Iden-ti-fi-ca-ti-on (SECSI 2010),* , 2010.

[72] E. Öztürk, G. Hammouri, and B. Sunar, "Towards Robust Low Cost Authentication for Pervasive Devices," in *2008 Sixth Annual IEEE International Conference on Pervasive Computing and Communications (PerCom)*, 2008, pp. 170–178.

[73] H. Martin, P. Peris-Lopez, G. Natale, M. Taouil, and S. Hamdioui, "Enhancing PUF Based Challenge–Response Sets by Exploiting Various Background Noise Configurations,"

Electronics, vol. 8, no. 2, p. 145, Jan. 2019.

[74] M. N. Aman, K. C. Chua, and B. Sikdar, "Mutual Authentication in IoT Systems Using Physical Unclonable Functions," *IEEE Internet Things J.*, vol. 4, no. 5, pp. 1327–1340, Oct. 2017.

[75] W. Adi, N. Ouertani, A. Hanoun, and B. Soudan, "Deploying FPGA self-configurable cell structure for micro crypto-functions," in *2009 IEEE Symposium on Computers and Communications*, 2009, pp. 348–354.

[76] M. Fyrbiak, C. Kison, M. Jeske, and W. Adi, "Combined HW-SW adaptive clone-resistant functions as physical security anchors," in *2013 NASA/ESA Conference on Adaptive Hardware and Systems (AHS-2013)*, 2013, pp. 130–137.

[77] Z. E. A. A. Ismaili and A. Moussa, "Self-Partial and Dynamic Reconfiguration Implementation for AES using FPGA," *Comput. Sci. Cryptogr. Secur. arXiv0909.2369*, Sep. 2009.

[78] R. P. S. Sidhu, A. Mei, and V. K. Prasanna, "Genetic Programming Using Self-Reconfigurable FPGAs," in *In: Lysaght P., Irvine J., Hartenstein R. (eds) Field Programmable Logic and Applications. FPL 1999. Lecture Notes in Computer Science, vol 1673.* , 1999, pp. 301–312.

[79] "SmartFusion2 SoC FPGAs | Microsemi." [Online]. Available: https://www.microsemi.com/product-directory/soc-fpgas/1692-smartfusion2. [Accessed: 08-Sep-2019].

[80] S. Mulhem, M. Mohammad, and W. Adi, "A New Low-Complexity Cipher Class for Clone-Resistant Identities," in *2019 42nd International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, 2019, pp. 971–976.

[81] R. Gennaro, A. Lysyanskaya, T. Malkin, S. Micali, and T. Rabin, "Algorithmic Tamper-Proof (ATP) Security: Theoretical Foundations for Security against Hardware Tampering," in *Theory of Cryptography. TCC 2004. Lecture Notes in Computer Science, vol 2951. Springer, Berlin, Heidelberg*, 2004, pp. 258–277.

[82] T. Wollinger, J. Guajardo, and C. Paar, "Security on FPGAs," *ACM Trans. Embed. Comput. Syst.*, vol. 3, no. 3, pp. 534–574, Aug. 2004.

[83] P. Y. A. Ryan, "Mathematical Models of Computer Security," in *Foundations of Security Analysis and Design. FOSAD 2000. Lecture Notes in Computer Science,* vol. 2171, Focardi R. and Gorrieri R., Eds. Springer, Berlin, Heidelberg, 2001, pp. 1–62.

[84] P. Swire, "A Model for When Disclosure Helps Security: What Is Different About Computer and Network Security?," *J. Telecommun. High Technol. Law* , vol. 163, p. 46, 2004.

[85] T. Wollinger, J. Guajardo, and C. Paar, "Security on FPGAs," *ACM Trans. Embed. Comput. Syst.*, vol. 3, no. 3, pp. 534–574, Aug. 2004.

[86] U. Maurer, K. Pietrzak, and R. Renner, "Indistinguishability Amplification," in *Advances in Cryptology - CRYPTO 2007*, Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 130–149.

[87] J. H. Kim, Nam Yong;Rathore, Shailendra;Ryu, Jung Hyun;Park, Jin Ho;Park, "A Survey on Cyber Physical System Security for IoT: Issues, Challenges, Threats, Solutions," *J. Inf. Process. Syst.*, vol. 14, no. 6, pp. 1361–1384, 2018.

[88] G. Nemes, "On the Coefficients of the Asymptotic Expansion of n!," *J. Integer Seq.*, vol. 13, no. 10.6.6, 2010.

[89] R. van den Berg, “Entropy analysis of physical unclonable functions,” Eindhoven University of Technology, 2012.

[90] M. A. Nielsen and I. L. Chuang, *Quantum computation and quantum information*. Cambridge University Press, 2010.

[91] V. Mavroeidis, K. Vishi, M. D. Zych, and A. Jøsang, “The Impact of Quantum Computing on Present Cryptography,” Mar. 2018.

[92] R. Jozsa, “Entanglement and Quantum Computation,” Jul. 1997.

[93] L. K. Grover and L. K., “A fast quantum mechanical algorithm for database search,” in *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing - STOC ’96*, 1996, pp. 212–219.

[94] J. Serrano, “Introduction to FPGA design,” in *CAS - CERN Accelerator School: Course on Digital Signal Processing*, 2007, pp. 231–247.

[95] U. Farooq, Z. Marrakchi, and H. Mehrez, “FPGA Architectures: An Overview,” in *Tree-based Heterogeneous FPGA Architectures*, New York, NY: Springer New York, 2012, pp. 7–48.

[96] S. M. Trimberger and J. J. Moore, “FPGA Security: Motivations, Features, and Applications,” *Proc. IEEE*, vol. 102, no. 8, pp. 1248–1265, Aug. 2014.

[97] T. Tuan, T. Strader, and S. Trimberger, “Analysis of Data Remanence in a 90nm FPGA,” in *2007 IEEE Custom Integrated Circuits Conference*, 2007, pp. 93–96.

[98] S. Kolay and D. Mukhopadhyay, “Khudra: A New Lightweight Block Cipher for FPGAs,” in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2014, vol. 8804, pp. 126–145.

[99] S. Mulhem, A. Mars, and W. Adi, “A Cipher Class Based on Golden S-Boxes for Creating Clone-Resistant Identities,” Springer, Cham, 2019, pp. 3–14.

[100] S. Mulhem, M. Ayache, and W. Adi, “Mini-Block-Based Cipher Class for Physical Clone-Resistant Devices,” in *EST - Eighth IEEE International Conference on Emerging Security Technologies*, 2019.

[101] G. Hatzivasilis, K. Fysarakis, I. Papaefstathiou, and C. Manifavas, “A review of lightweight block ciphers,” *J. Cryptogr. Eng.*, vol. 8, no. 2, pp. 141–184, Jun. 2018.

[102] K. Aoki *et al.*, “Camellia: A 128-Bit Block Cipher Suitable for Multiple Platforms — Design andAnalysis,” in *SAC 2000. Lecture Notes in Computer Science, vol 2012*, 2001, pp. 39–56.

[103] W. Wu and L. Zhang, “LBlock: A Lightweight Block Cipher,” in *Applied Cryptography and Network Security. ACNS 2011. Lecture Notes in Computer Science, vol 6715.* , 2011, pp. 327–344.

[104] K. Shibutani, T. Isobe, H. Hiwatari, A. Mitsuda, T. Akishita, and T. Shirai, “Piccolo: An Ultra-Lightweight Blockcipher,” in *Cryptographic Hardware and Embedded Systems – CHES 2011. CHES 2011. Lecture Notes in Computer Science, vol 6917.*, 2011, pp. 342–357.

[105] M. Izadi, B. Sadeghiyan, S. S. Sadeghian, and H. A. Khanooki, “MIBS: A New Lightweight Block Cipher,” in *Cryptology and Network Security. CANS 2009. Lecture Notes in Computer Science, vol 5888*, 2009, pp. 334–348.

[106] R. Beaulieu, D. Shors, J. Smith, S. Treatman-Clark, B. Weeks, and L. Wingers, “The SIMON and SPECK lightweight block ciphers,” in *Proceedings of the 52nd Annual Design Automation Conference on - DAC ’15*, 2015, pp. 1–6.

[107] W. Diffie and M. E. Hellman, "Exhaustive Cryptanalysis of the NBS Data Encryption Standard," *Computer (Long. Beach. Calif).*, vol. 10, no. 6, pp. 74–84, Jun. 1977.

[108] M. J. Wiener, "Efficient DES key search," 1994.

[109] E. Biham and A. Shamir, "Differential cryptanalysis of DES-like cryptosystems," *J. Cryptol.*, vol. 4, no. 1, pp. 3–72, 1991.

[110] E. Biham and A. Shamir, *Differential Cryptanalysis of the Data Encryption Standard*. New York, NY: Springer New York, 1993.

[111] M. Matsui, "Linear Cryptanalysis Method for DES Cipher," Springer, Berlin, Heidelberg, 1994, pp. 386–397.

[112] M. Luby and C. Rackoff, "How to Construct Pseudorandom Permutations from Pseudorandom Functions," *SIAM J. Comput.*, vol. 17, no. 2, pp. 373–386, Apr. 1988.

[113] V. Nachef, J. Patarin, and E. Volte, *Feistel Ciphers*. Cham: Springer International Publishing, 2017.

[114] U. M. Maurer, "A Simplified and Generalized Treatment of Luby-Rackoff Pseudorandom Permutation Generators," in *Advances in Cryptology — EUROCRYPT' 92*, 1992, pp. 239–255.

[115] S. Patel, Z. Ramzan, and G. S. Sundaram, "Luby-Racko. Ciphers: Why XOR Is Not So Exclusive," in *Selected Areas in Cryptography. SAC 2002. Lecture Notes in Computer Science, vol 2595.*, 2003, pp. 271–290.

[116] J. Patarin, "Generic Attacks on Feistel Schemes," in *Advances in Cryptology — ASIACRYPT 2001. ASIACRYPT 2001. Lecture Notes in Computer Science, vol 2248*, 2001, pp. 222–238.

[117] J. Patarin, "Luby-Rackoff: 7 Rounds Are Enough for 2 n(1 − ε) Security," in *Advances in Cryptology - CRYPTO 2003. CRYPTO 2003. Lecture Notes in Computer Science, vol 2729.*, 2003, pp. 513–529.

[118] J. Patarin, "About Feistel Schemes with Six (or More) Rounds," in *Fast Software Encryption. FSE 1998. Lecture Notes in Computer Science, vol 1372.*, 1998, pp. 103–121.

[119] J. Patarin, "New Results on Pseudorandom Permutation Generators Based on the Des Scheme," in *Advances in Cryptology — CRYPTO '91*, 1991, pp. 301–312.

[120] Y. Zheng, T. Matsumoto, and H. Imai, "Impossibility and Optimality Results on Constructing Pseudorandom Permutations," in *Advances in Cryptology — EUROCRYPT '89*, 1989, pp. 412–422.

[121] J. Pieprzyk, "How to Construct Pseudorandom Permutations from Single Pseudorandom Functions," in *Lecture Notes in Computer Science*, 1991, pp. 140–150.

[122] M. Naor and O. Reingold, "On the Construction of Pseudorandom Permutations: Luby—Rackoff Revisited," *J. Cryptol.*, vol. 12, no. 1, pp. 29–66, Jan. 1999.

[123] G. Carter, E. Dawson, and L. Nielsen, "DESV: A Latin Square variation of DES," in *Workshop on Selected Areas of Cryptography*, 1995, pp. 158–172.

[124] S. Patel, Z. Ramzan, and G. S. Sundaram, "Luby-Rackoff Ciphers over Finite Algebraic Structures or Why XOR is not so Exclusive," in *Selected Areas in Cryptography - SAC 2002, LNCS 2595.* , 2002, pp. 271–290.

[125] R. L. Rivest, "Permutation Polynomials Modulo 2w," *Finite Fields Their Appl.*, vol. 7, no. 2, pp. 287–292, Apr. 2001.

[126] A. Klimov and A. Shamir, "A new class of invertible mappings," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 2523, pp. 470–

483, 2003.

[127] G. Leander and A. Poschmann, “On the Classification of 4 Bit S-Boxes,” in *rithmetic of Finite Fields. WAIFI 2007. Lecture Notes in Computer Science, vol 4547.*, 2007, pp. 159–176.

[128] M.-J. O. Saarinen, “Cryptographic Analysis of All 4 × 4-Bit S-Boxes,” in *Selected Areas in Cryptography. SAC 2011. Lecture Notes in Computer Science, vol 7118.*, 2012, pp. 118–133.

[129] R. L. Rivest, M. J. B. Robshaw, R. Sidney, and Y. L. Yin, “The RC 6 TM Block Cipher,” 1998.

[130] Y. Zheng, T. Matsumoto, and H. Imai, “On the Construction of Block Ciphers Provably Secure and Not Relying on Any Unproved Hypotheses,” in *Advances in Cryptology — CRYPTO’ 89 Proceedings. Lecture Notes in Computer Science, vol 435.*, 1989, pp. 461–480.

[131] T. Suzaki and K. Minematsu, “Improving the Generalized Feistel,” 2010, pp. 19–39.

[132] V. T. Hoang and P. Rogaway, “On Generalized Feistel Networks,” in *Advances in Cryptology – CRYPTO 2010. Lecture Notes in Computer Science, vol 6223. Springer, Berlin, Heidelberg.*, 2010, vol. 6223, pp. 613–630.

[133] M. Sajadieh, A. Mirzaei, H. Mala, and V. Rijmen, “A new counting method to bound the number of active S-boxes in Rijndael and 3D,” *Des. Codes Cryptogr.*, vol. 83, no. 2, pp. 327–343, May 2017.

[134] C. Rolfes, A. Poschmann, G. Leander, and C. Paar, “Ultra-Lightweight Implementations for Smart Devices – Security for 1000 Gate Equivalents,” in *mart Card Research and Advanced Applications. CARDIS 2008. Lecture Notes in Computer Science, vol 5189.*, 2008, pp. 89–103.

[135] J. Borghoff, “Cryptanalysis of Lightweight Ciphers,” Technical University of Denmark (DTU), 2011.

[136] J. Borghoff, L. R. Knudsen, G. Leander, and S. S. Thomsen, “Cryptanalysis of PRESENT-Like Ciphers with Secret S-Boxes,” in *Fast Software Encryption. FSE 2011. Lecture Notes in Computer Science, vol 6733.*, 2011, pp. 270–289.

[137] A. Klimov and A. Shamir, “Cryptographic Applications of T-Functions,” Springer, Berlin, Heidelberg, 2004, pp. 248–261.

[138] Y. Liu, V. Rijmen, and G. Leander, “Nonlinear diffusion layers,” *Des. Codes Cryptogr.*, vol. 86, no. 11, pp. 2469–2484, Nov. 2018.

[139] L. Cui, L. Cui, and Y. Cao, “A NEW S-BOX STRUCTURE NAMED AFFINE-POWER-AFFINE,” *Int. J. Innov. Inf. Control*, vol. 3, no. 3, pp. 751–759, 2007.

[140] M. T. Tran, D. K. Bui, and A. D. Duong, “Gray S-Box for Advanced Encryption Standard,” in *2008 International Conference on Computational Intelligence and Security*, 2008, pp. 253–258.

[141] E. S. Abuelyman, A.-A. S. Alsehibani, and S. Arabia, “An Optimized Implementation of the S-Box using Residues of Prime Numbers,” *IJCSNS Int. J. Comput. Sci. Netw. Secur.*, vol. 8, no. 4, 2008.

[142] A. Bogdanov *et al.*, “PRESENT: An Ultra-Lightweight Block Cipher,” in *Cryptographic Hardware and Embedded Systems - CHES 2007*, Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 450–466.

[143] R. P. Singh and S. Maity, “Permutation Polynomials modulo pn,” *undefined*, 2009.

[144] J. Ryu and O. Y. Takeshita, “On Inverses for Quadratic Permutation Polynomials over Integer Rings,” Feb. 2011.

[145] J. Ryu and O. Y. Takeshita, “On quadratic inverses for quadratic permutation polynomials over integer rings,” *IEEE Trans. Inf. Theory*, vol. 52, no. 3, pp. 1254–1260, Mar. 2006.

[146] H. Tapia-Recillas, “Remarks on Self-Inverse Quadratic Permutation Polynomials,” *Int. J. Algebr.*, vol. 4, no. 19, pp. 931–938, 2010.

[147] A. Klimov, “Applications of T-functions in Cryptography,” Weizmann Institute of Science, 2005.

[148] G. Keller and F. R. Olson, “Counting polynomial functions $(\mod p^n)$,” *Duke Math. J.*, vol. 35, no. 4, pp. 835–838, Dec. 1968.

[149] Q. Zhang, “Polynomial functions and permutation polynomials over some finite commutative rings,” *J. Number Theory*, vol. 105, no. 1, pp. 192–202, Mar. 2004.

[150] J. J. Jiang, “On the number counting of polynomial functions,” *J. Math. Res. Expo.*, vol. 30, no. 2, pp. 241–248, 2010.

[151] S. Li, “Null Polynomials modulo m,” *arXiv Prepr. math/0510217*, Oct. 2005.

[152] T. Jakobsen and L. R. Knudsen, “The interpolation attack on block ciphers,” Springer, Berlin, Heidelberg, 1997, pp. 28–40.

[153] T. Jakobsen, “Cryptanalysis of block ciphers with probabilistic non-linear relations of low degree,” Springer, Berlin, Heidelberg, 1998, pp. 212–222.

[154] J. Steinberger, “Improved Security Bounds for Key-Alternating Ciphers via Hellinger Distance.” Cryptology ePrint Archive, Report 2012/481, 2012.

[155] R. Lampe, J. Patarin, and Y. Seurin, “An Asymptotically Tight Security Analysis of the Iterated Even-Mansour Cipher,” in *Advances in Cryptology – ASIACRYPT 2012. ASIACRYPT 2012. Lecture Notes in Computer Science, vol 7658.*, 2012, pp. 278–295.

[156] S. Chen and J. Steinberger, “Tight Security Bounds for Key-Alternating Ciphers,” in *Advances in Cryptology – EUROCRYPT 2014. EUROCRYPT 2014. Lecture Notes in Computer Science, vol 8441.*, 2014, pp. 327–350.

[157] Z. Yan, P. Zhang, and A. V. Vasilakos, “A survey on trust management for Internet of Things,” *J. Netw. Comput. Appl.*, vol. 42, pp. 120–134, Jun. 2014.

[158] Fenye Bao and Ing-Ray Chen, “Trust management for the internet of things and its application to service composition,” in *2012 IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, 2012, pp. 1–6.

[159] M. Hamad, M. Nolte, and V. Prevelakis, “A framework for policy based secure intra vehicle communication,” in *2017 IEEE Vehicular Networking Conference (VNC)*, 2017, pp. 1–8.

[160] P. K. Verma *et al.*, “Machine-to-Machine (M2M) communications: A survey,” *J. Netw. Comput. Appl.*, vol. 66, pp. 83–105, May 2016.

[161] Y.-K. Chen, “Challenges and opportunities of internet of things,” in *17th Asia and South Pacific Design Automation Conference*, 2012, pp. 383–388.

[162] M. Burhan, R. Rehman, B. Khan, and B.-S. Kim, “IoT Elements, Layered Architectures and Security Issues: A Comprehensive Survey,” *Sensors*, vol. 18, no. 9, p. 2796, Aug. 2018.

[163] S. Mulhem, W. Adi, A. Mars, and V. Prevelakis, “Chaining trusted links by deploying secured physical identities,” in *2017 Seventh International Conference on Emerging Security Technologies (EST)*, 2017, pp. 215–220.

[164] W. Adi, S. Mulhem, and A. Mars, “Secured remote sensing by deploying clone-resistant

Secret Unknown Ciphers," in *2017 IEEE International Conference on Consumer Electronics - Taiwan (ICCE-TW)*, 2017, pp. 133–134.

[165] G. Gan, Z. Lu, and J. Jiang, "Internet of Things Security Analysis," in *2011 International Conference on Internet Technology and Applications*, 2011, pp. 1–4.

[166] J. Granjal, E. Monteiro, and J. Sa Silva, "Security for the Internet of Things: A Survey of Existing Protocols and Open Research Issues," *IEEE Commun. Surv. Tutorials*, vol. 17, no. 3, pp. 1294–1312, 2015.

[167] M. Hamad, M. Nolte, and V. Prevelakis, "Towards Comprehensive Threat Modeling for Vehicles," in *CERTS 2016, 1st Workshop on Security and Dependability of Critical Embedded Real-Time Systems*, 2016.

[168] D. G. Padmavathi and M. D. Shanmugapriya, "A Survey of Attacks, Security Mechanisms and Challenges in Wireless Sensor Networks," *Int. J. Comput. Sci. Inf. Secur.*, vol. 4, Sep. 2009.

[169] A. C. Sarma and J. Girão, "Identities in the Future Internet of Things," *Wirel. Pers. Commun.*, vol. 49, no. 3, pp. 353–363, May 2009.

[170] S. G. Weber, L. Martucci, S. Ries, and M. Mühlhäuser, "Towards trustworthy identity and access management for the future internet," in *4th International Workshop on Trustworthy Internet ofPeople, Things & Services*, 2010.

[171] H. Vogt, "Small Worlds and the Security of Ubiquitous Computing," in *Sixth IEEE International Symposium on a World of Wireless Mobile and Multimedia Networks*, pp. 593–597.

[172] I. Sohn, "Small-World and Scale-Free Network Models for IoT Systems," *Mob. Inf. Syst.*, vol. 2017, pp. 1–9, Jan. 2017.

[173] B. Halak, M. Zwolinski, and M. S. Mispan, "Overview of PUF-based hardware security solutions for the internet of things," in *2016 IEEE 59th International Midwest Symposium on Circuits and Systems (MWSCAS)*, 2016, pp. 1–4.

[174] S. Mulhem, A. Abadleh, and W. Adi, "Accelerometer-Based Joint User-Device Clone-Resistant Identity," in *2018 Second World Conference on Smart Trends in Systems, Security and Sustainability (WorldS4)*, 2018, pp. 230–237.

[175] E. Hamadaqa, S. Mulhem, A. Mars, and W. Adi, "Clone-Resistant Joint-Identity Technique for Securing Fleet Management Systems," in *2018 NASA/ESA Conference on Adaptive Hardware and Systems (AHS)*, 2018, pp. 327–332.

[176] R. Zarrouk, S. Mulhem, L. Reich, and W. Adi, "Non-Repeatable Clone-Resistant Group Device Identity," in *2019 International Conference on Cyber Security for Emerging Technologies (CSET)*, 2019, pp. 1–7.

[177] S. Vaudenay, "On Privacy Models for RFID," in *Advances in Cryptology – ASIACRYPT 2007. ASIACRYPT 2007. Lecture Notes International Conference on the Theory and Application of Cryptology and Information Security, in Computer Science, vol 4833*, 2007, pp. 68–87.

[178] K. Bosworth, M. G. Gonzalez Lee, S. Jaweed, and T. Wright, "Entities, identities, identifiers and credentials — what does it all mean?," *BT Technol. J.*, vol. 23, no. 4, pp. 25–36, Oct. 2005.

[179] A. Braeken, "PUF Based Authentication Protocol for IoT," *Symmetry (Basel).*, vol. 10, no. 8, p. 352, Aug. 2018.

DNA	Deoxyribonucleic Acid.
eDNA	Electronic DNA.
NVM	non-volatile memory.
PUFs	Physical Unclonable Functions.
PCRI	Physically Clone-Resistant Identity.
PRF	Pseudorandom function.
PRP	Pseudorandom Permutation.
PRNG	Pseudorandom Generator.
TRNG	True Random Number Generator.
POWF	Physical One-way Function.
CRP	Challenge-Response Pair.
IoT	Internet of things.
ML	Machine Learning.
IC	Integrated Circuit.
CA	Cloning Attack.
SCA	Side-channel Attack.
ECC	Error-Correction Code.
DB	Database.
HDA	Helper Data Algorithm.
PUPRF	Physical Unclonable Pseudorandom Function.
SPN	Substitution-Permutation Network.
AES	Advanced Encryption Standard.
DES	Data Encryption Standard.
SoC	System on Chip.
FPGAs	Field Programmable Gate Arrays.
SUC	Secret Unknown Cipher.
TA	Trusted Authority.
CPA	Adaptive Chosen Plaintext Attack.
nCPA	Non-Adaptive Chosen Plaintext Attack.
CCA	Adaptive Chosen Ciphertext Attack.

nCCA	Non-Adaptive Chosen Ciphertext Attack.
WCS	Worst-Case Scenario.
GS	Golden S-Boxes.
BP	Bundle Permutation.
GFN	Generalized Feistel Networks.
QPF	Quadratic Polynomial Function.
SIQPF	Self-Inverse Quadratic Polynomial Function.
TR	Trust Relationship.
IT	Identity Trust.
TCL	Trusted Communication Link.
SSK	Shared Session Key.
MITM	Man-in-the-middle Attack.